# ALGORITMOS DE CLUSTERING PARA APOYO AL DIAGNÓSTICO DE CELIAQUÍA

# CLUSTERING ALGORITHMS TO SUPPORT THE DIAGNOSIS OF CELIAC DISEASE

Carla Martínez Nieto-Márquez
Pablo Sanz Caperote

Doble grado en Ingeniería Informática y Matemáticas
Universidad Complutense de Madrid

---


Trabajo Fin de Grado

14 de junio de 2021

Tutores

Mercedes García Merayo
Manuel Núñez García

# Resumen en castellano

La Medicina y la Inteligencia Artificial están fuertemente ligadas. En los últimos años, se han utilizado las técnicas disponibles en la segunda disciplina para ayudar en el diagnóstico y tratamiento de enfermedades o para crear herramientas de apoyo a los profesionales médicos.

El objetivo de este trabajo es ayudar al diagnóstico de la enfermedad celíaca en pacientes que tienen una clínica poco habitual y encontrar patrones entre la clínica de estos pacientes. Para ello trabajaremos con datos cedidos por la doctora Concepción Núñez Pardo de Vera, sobre los cuales haremos un importante trabajo de formateo, aplicaremos algoritmos de clustering y evaluaremos los resultados obtenidos.

## Palabras clave

Celiaquía, Inteligencia artificial, Algoritmos de clustering

# Abstract

Medicine and Artificial Intelligence are strongly linked. In the last few years, the techniques available in the second named discipline have been used to help with the diagnosis and treatment of illnesses or to create tools for the backing of medical experts.

The objective of this project is to help in the diagnosis of Celiac Disease in patients with unusual clinic records. For doing so we will work with data given by the doctor Concepción Núnez Pardo de Vera, we will format this data, apply clustering algorithms to it and evaluate the obtained results.

## Keywords

Celiac Disease, Artificial Intelligence, Clustering Algorithms

# Índice general

**Índice** I

**1. Introducción** 1
- 1.1. Motivación . . . 1
- 1.2. Objetivo . . . 1
- 1.3. Estructura de la memoria . . . 2

**1. Introduction** 3
- 1.1. Motivation . . . 3
- 1.2. Objectives . . . 3
- 1.3. Work Structure . . . 4

**2. Formateo de los datos** 5
- 2.1. Estudio del dataset . . . 5
- 2.2. Tratamiento de los datos categóricos . . . 11
- 2.3. Datos numéricos incompletos . . . 12
  - 2.3.1. Eliminación de datos . . . 12
  - 2.3.2. Imputación . . . 13
- 2.4. Escalado de los datos . . . 15
- 2.5. Datasets generados . . . 18

**3. Base teórica de los algoritmos de clústering** 21
- 3.1. Algoritmos de partición . . . 21
  - 3.1.1. $K$-means . . . 22
  - 3.1.2. $K$-prototypes . . . 23
  - 3.1.3. $K$-modes . . . 25
  - 3.1.4. $K$-POD . . . 25
  - 3.1.5. $K$-medoids . . . 27
- 3.2. Clústering aglomerativo . . . 28
- 3.3. Clústering espectral . . . 31
- 3.4. Algoritmos basados en densidad . . . 33
  - 3.4.1. DBSCAN . . . 33
  - 3.4.2. OPTICS . . . 35

**4. Base teórica de los métodos de evaluación** 39
- 4.1. Evaluación del dataset . . . 39
- 4.2. Evaluación de los métodos de clústering . . . 40

4.2.1. Métodos intrínsecos . . . 40
4.2.2. Métodos extrínsecos . . . 41

**5. Resultados** **45**
5.1. Pruebas previas . . . 45
5.1.1. Fijando el número de clústeres a 2 . . . 45
5.1.2. Mejorando la imputación . . . 46
5.2. Evaluación de los datasets . . . 47
5.3. Evaluación de los métodos . . . 48
5.3.1. K-Means . . . 48
5.3.2. K-Prototypes . . . 49
5.3.3. K-Modes . . . 50
5.3.4. K-POD . . . 50
5.3.5. K-Medoids . . . 50
5.3.6. Espectral . . . 51
5.3.7. Aglomerativo . . . 52
5.3.8. DBSCAN . . . 53
5.3.9. OPTICS . . . 53
5.3.10. Comparación de la distribución en clústeres . . . 54

**6. Contribuciones individuales al proyecto** **57**
6.1. Contribuciones de Carla Martínez Nieto-Márquez . . . 57
6.2. Contribuciones de Pablo Sanz Caperote . . . 60

**7. Conclusión y trabajo futuro** **63**
7.1. Calidad de los datos . . . 63
7.2. Métodos de clustering . . . 64

**7. Conclusions and Future work** **65**
7.1. Data quality . . . 65
7.2. Clustering algorithms . . . 65

**Bibliografía** **69**

**A. Repositorio y ejecución en local** **71**
A.1. Descarga y ejecución en local . . . 71
A.2. Estructura del repositorio . . . 73

**B. Elementos gráficos** **75**
B.1. Coeficiente de Silhouette para un algoritmo y datasets concretos . . . 76
B.2. Tabla estadístico de Hopkins . . . 77
B.3. Gráfico interactivo para un algoritmo y datasets concretos . . . 78
B.4. Tabla clasificatoria de los pacientes . . . 79
B.5. Tabla resumen para un algoritmo . . . 80

# Capítulo 1

# Introducción

En este capítulo presentamos brevemente la motivación detrás del desarrollo de este Trabajo de Fin de Grado, el objetivo que queremos alcanzar y la estructura del resto de la memoria.

## 1.1. Motivación

Durante estos últimos años hemos podido observar cómo el desarrollo de la Informática, y en especial de la Inteligencia Artificial, ha supuesto un cambio importante en muchos ámbitos de nuestras vidas. La aplicación de las herramientas que nos aporta esta nueva área nos dan la posibilidad de desarrollar importantes proyectos impensables hace no tanto tiempo, ya que dichas herramientas nos permiten manejar grandes volúmenes de datos, encontrar patrones imperceptibles al razonamiento humano, o simplemente tratar de reforzar una intuición tras haber hecho un estudio de los datos. A raíz de todo esto surgió la idea de realizar un proyecto para mejorar el diagnóstico de la enfermedad celiaca en colaboración con el Hospital Clínico San Carlos y la médica Concepción Núñez Pardo de Vera y su equipo del área de Digestivos. El trabajo se engloba dentro de una nueva línea de investigación para el diagnóstico y la mejora de la calidad de vida de los pacientes que sufren dicha enfermedad, que aproximadamente son un 1 % de la población mundial según la OMS.

## 1.2. Objetivo

El objetivo del proyecto es continuar con la línea de investigación ya empezada en el año 2020 [26] entre la Facultad de Informática de la Universidad Complutense de Madrid y el Hospital Clínico San Carlos. Más concretamente, el objetivo es ayudar al diagnóstico de la enfermedad celiaca en pacientes que no cursan una clínica habitual.

Se partió de una base de datos de pacientes con multitud de campos que había sido rellenada tras el estudio de cada uno de los pacientes por parte del equipo de Digestivos del hospital. En esta base de datos se podían diferenciar, a grandes rasgos, tres tipos de pacientes: diagnosticados que sufrían la enfermedad celiaca, diagnosticados que no la padecían,

y los que en el momento del estudio no se sabía. Por lo tanto, los objetivos finales que se presentaron en la elaboración del proyecto eran los siguientes:

- Encontrar algún patrón para los pacientes que habían sido diagnosticados como celiacos o como no celiacos.
- Encontrar algún patrón que permitiera establecer un diagnóstico para los pacientes que no habían sido diagnosticados.

En ambos casos decidimos que lo más adecuado era utilizar técnicas de clústering. Obviamente, la información de diagnóstico solo fue usada para la evaluación de resultados y no para la clasificación de los pacientes en la búsqueda de patrones.

## 1.3. Estructura de la memoria

El resto de la memoria en la que se presenta este Trabajo de Fin de Grado está dividida en los siguientes capítulos:

- *Formateo de los datos.* Se explica todo el proceso que se llevó a cabo desde la obtención de los datos hasta el momento en que estuvieron listos para introducirlos en los modelos de clustering.
- *Base teórica de los algoritmos utilizados.* Se enumeran y se detallan los diferentes algoritmos de clústering que se emplearán para obtener los resultados.
- *Base teórica de los métodos de evaluación.* Se enumeran y se detallan los diferentes métodos empleados para medir el resultado de los métodos de clústering.
- *Resultados.* Se exponen los resultados obtenidos tras la aplicación de cada uno de los métodos.
- *Contribuciones individuales al proyecto.* Se muestran las contribuciones de cada uno de nosotros.
- *Conclusiones y trabajos futuros.* Se exponen las conclusiones obtenidas del proyecto y se proponen ideas que consideramos que lo mejorarían para posibles trabajos futuros.
- *Repositorio y ejecución en local.* Se explica cómo se puede ejecutar el código de este proyecto en local y la estructura del repositorio que se puede encontrar en: `https://github.com/TFG-Informatica-Enfermedad-Celiaca/Analisis-EC`.
- *Elementos gráficos.* Se muestra un pequeño ejemplo de las gráficas que genera nuestro programa.

# Capítulo 1

# Introduction

In this chapter we will briefly present what has motivated this project, the objectives and the structure of the paper.

## 1.1. Motivation

In recent times we have seen how Computer Science, and especially Artificial Intelligence have changed and deeply affected our lives. The techniques available in these areas allow us to develop important projects that weren't possible years ago. This is due to the fact that we are now able to work with bigger volumes of data, find imperceptible patterns to the human eye or reinforce human intuitions.

All this motivated the creation of a project to improve the diagnosis of Celiac disease in collaboration with Hospital Clínico San Carlos and the doctor Concepción Núñez Pardo de Vera and her team which is focused in the digestive area. This project is part of a new line of research studying the diagnostic and the improvement of life quality for the patients suffering this disease which according to the OMS represents the 1 % of the global population.

## 1.2. Objectives

The objective of this project is to continue the line of research started in the year 2020 [26] between the Computer Science Faculty of the Complutense University in Madrid and the Hospital Clínico San Carlos. Specifically the aim of this project is to help with the diagnosis of Celiac Disease in patients with unusual clinical records.

We started working with a database containing an enormous amount of fields (not as many rows) and that was filled by the digestive team in Hospital Clínico San Carlos. In this database we could easily differentiate three groups of patients, the ones diagnosed with the disease, the ones diagnosed without the disease (this includes allergies to gluten) and the patients without a diagnostic.

To sum up, the final objectives of the project were:

- To find some pattern for those patients that had been diagnosed (for both with the celiac disease and without).
- To find some pattern that allows us to diagnose the patients without a diagnostic.

In both cases we thought that the best option was to use clustering techniques as we have a lot of patients without labels, and many of the labels weren't reliable as said by the doctor. Obviously the information about the diagnostic was only used to evaluate the results and wasn't used in the search of patterns.

## 1.3. Work Structure

This paper is divided in the following chapters:

- Data formatting: Where we explain the process since we first obtain the data until it was ready to introduce in the clustering models.
- Theoretical basis of the clustering algorithms: Where we explain in detail each of the algorithms that were used to obtain the results.
- Theoretical basis of evaluation methods: Where we explain in detail each of the methods that were used to evaluate the results of the clustering methods.
- Results: We present the results gotten after applying each of the methods.
- Individual contributions to the project: Where we explain the work done by each of the components in the group.
- Conclusions and future work: We expone the conclusions obtained after finishing the project and we propose some ideas to improve the results.
- *Repository and local execution.* We explain how to execute the project in local and the structure of the repository: https://github.com/TFG-Informatica-Enfermedad-Celiaca/Analisis-EC.
- Graphic elements: Where we show a little example of the graphs and tables that are generated by our program.

# Capítulo 2

# Formateo de los datos

En este primer capítulo se muestra el proceso que se ha llevado a cabo para el formateo de los datos, desde la obtención de los mismos hasta el momento en el que han estado preparados para introducirlos en los diferentes algoritmos de clustering. Este proceso ha consistido en varias fases que explicaremos a continuación. La primera de estas fases fue el estudio de los datos, seguido de la eliminación de columnas redundantes e innecesarias, continuado con la completación de casillas vacías, para terminar con la transformación de los datos para su correcto uso en los diferentes algoritmos.

## 2.1. Estudio del dataset

Antes de adentrarnos en la preparación de los datos para su posterior estudio mediante la aplicación de algoritmos de clustering, tuvimos que estudiarlos y conocerlos bien. Por pertenecer estos datos a un ámbito totalmente desconocido para nosotros, el ámbito médico y más en concreto el ámbito de la enfermedad celiaca, este proceso supuso un gran esfuerzo e inversión de tiempo. Además de realizar un estudio preliminar, a través de paginas web especializadas en celiaquía [7; 22], para poder comprender el significado de los datos, tuvimos una reunión con la médica que nos proporcionó el dataset. En esta reunión pudimos aclarar nuestras dudas y así entender mejor los datos con los que íbamos a trabajar para posteriormente poder tomar decisiones informadas durante la fase del procesamiento de los datos.

Nuestro proyecto se enmarca en la búsqueda de avances para el diagnóstico de la enfermedad celiaca desarrollado en pacientes que no tienen un diagnóstico claro. Esta es una línea de trabajo llevada a cabo por la médica Concepción Núñez Pardo de Vera y su equipo del área de Digestivo en el Hospital Clínico San Carlos. El dataset contiene datos que han sido recogidos por la médica durante las consultas y contiene un total de 96 columnas. Hay que tener en cuenta que no todas ellas son relevantes para nuestro trabajo y que muchas de ellas estaban prácticamente vacías. De este modo, a la vez que recibimos el dataset con la base de datos de todos los pacientes, recibimos un listado con 73 columnas *importantes* que la médica consideró que tenían especial relevancia.

Después de la reunión con la médica, ya disponíamos de la información necesaria para empezar a formatear nuestros datos. Lo primero que hicimos fue un listado de las columnas relevantes, indicando por cada una cuál era su significado y el tratamiento que le daríamos.

- *Diagnóstico.* Este campo puede reflejar varios diagnósticos: si el paciente padece la enfermedad; si según las pruebas no la padece, pero al eliminar el gluten de su dieta su clínica mejora o incluso desaparece, o si por el contrario ni aun eliminándolo desaparecen los síntomas; si tiene ciertas patologías propia de la celiaquía pero no otras; y, por último, si no padece la enfermedad. Esta columna solo se utilizará para la evaluación de los diferentes resultados, ya que no tendría ningún sentido utilizarla para la formación de los clústeres.

- *Indique país de origen o en su defecto la información disponible.* El país de origen es un buen indicador para conocer la genética del individuo. Y conocer este dato es importante ya que se está probado que la enfermedad celiaca depende en cierta medida de la genética de los pacientes. Nuestra base de datos está formada en su mayoría por personas europeas y por lo tanto, así distinguiremos los casos: personas europeas y Otros.

- *Sexo.* El género de los enfermos de celiaquía es básico pues existen diferencias en el desarrollo de la enfermedad entre hombres y mujeres [21].

- *Edad del paciente.* La médica nos indicó que debíamos eliminar la información perteneciente a pacientes menores de 18 años porque su clínica es muy distinta a la de los adultos, y que los pacientes mayores de 70 años tenían especial interés porque la clínica era totalmente distinta. Finalmente, determinó que una buena división en franjas sería de 10 en 10 años.

- *Grado de parentesco, Grado de parentesco (si hay más de 1).* Estos campos contienen es el grado de parentesco con el familiar que padece la enfermedad celiaca. Consideramos que es interesante conocer cuantos más antecedentes familiares mejor. Por este motivo crearemos cuatro columnas binarias, una por cada grado de parentesco, que indican si el paciente tiene o no un familiar afectado con ese grado.

- *Enfermedad inmunológica, Enfermedad inmunológica (si hay más de 1), Enfermedad inmunológica (si hay más de 2).* Estos campos contienen las enfermedades que sufre el paciente. A pesar de que la médica nos indicó que existe una lista con todos los posibles valores de esta columna, puede ser que no aparezcan en nuestro dataset todos los posibles valores con lo que crearíamos columnas rellenas únicamente de ceros, y podría ser que se ampliara esta lista en un futuro. Por estos motivos lo que haremos es crear una columna binaria por cada una de las enfermedades inmunológicas que aparecen y asignar el valor 1 a los pacientes que sufren dicha enfermedad.

- *Síntomas específicos, Síntomas específicos.1, Síntomas específicos.2.* Estos campos contienen los síntomas que han percibido los pacientes. La médica nos dijo que al igual

que ocurre con las enfermedades inmunológicas, estos campos están tipificados y por los mismos motivos que en el caso anterior crearemos una columna binaria para cada uno de los elementos que aparecen en estas columnas.

- *Signos, Signos 2, Signos 3.* Los signos clínicos son las manifestaciones objetivas, clínicamente fiables y observadas en la exploración médica, como por ejemplo la anemia. Estos campos los trataremos de la misma forma que hemos indicado para los campos correspondientes a enfermedades inmunológicas y síntomas.

- *HLA: grupos de riesgo, Haplotipo1, Haplotipo2.* Para entender estas tres columnas es necesario saber que la enfermedad celiaca está directamente asociada con los genes localizados en la región HLA (*Human Leukocyte Antigen*). A su vez, los genes de un paciente dependen de los genes de cada uno de los padres Haplotipo1 y Haplotipo2. En la columna *HLA: grupos de riesgo* aparecen los valores para el HLA (que se calculan a partir del valor de los Haplotipos de los progenitores) o bien se califica como "sin riesgo" en caso de que los dos campos de haplotipos sean de este tipo. En función del valor del HLA se sufre un mayor o menor riesgo de padecer la enfermedad. De mayor a menor riesgo tenemos las siguientes categorías [25]:

  1. DQ2.5 doble dosis: dos haplotipos con valor DQ2.5, o un haplotipo con valor DQ2.2 y otro con valor DQ2.5.
  2. DQ2.5 una dosis y DQ8 doble dosis: para tener HLA DQ2.5 una dosis es necesario tener un haplotipo con valor DQ2.5, o un haplotipo con valor DQ7.5 y otro con valor DQ2.2. Para tener HLA DQ8 es necesario tener ambos Haplotipos con valor DQ8.
  3. DQ8 una dosis: uno de los haplotipos con valor DQ8.
  4. DQ2.2. DQ7.5: para tener HLA DQ2.2 es necesario tener al menos un haplotipo DQ2.2 (y en caso de que solo sea uno, que el otro haplotipo sea sin riesgo), lo mismo ocurre con DQ7.5
  5. Sin riesgo.

  Lo que haremos con estas tres columnas es unificarlas en una única columna categórica, ya que, como hemos dicho, el campo HLA está determinado por los campos de los haplotipos y por tanto podemos prescindir de la información redundante. Pero antes de deshacernos de las columnas de los haplotipos tuvimos que rellenar correctamente la columna de HLA pues había un número elevado de pacientes que tenía información en los campos de los haplotipos pero no tenía información para HLA.

- *DCG_ATG2_1, DCG_ATG2_2.* Estas columnas se refieren al resultado de la misma prueba pero realizada en momentos distintos. Por lo tanto, reciben el mismo tratamiento. Esta prueba consiste en medir el título del anticuerpo ATG (anticuerpo anitransglutaminasa) cuando el paciente está siguiendo una dieta con Gluten (DCG). Los valores de esta columna se pueden obtener simplemente mirando los datos de las

columnas *Indicar título del anticuerpo*, que tienen valores numéricos. En caso de que el valor numérico en esas columnas sea menor que 20 añadimos en la columna *DCG_ATG* el valor "Negativo" y si es mayor que 20, "Positivo". Otro caso común es que mientras en las columnas *DCG_ATG* tengamos valores, las columnas numéricas estén vacías. En estos casos imputaremos un valor y comprobaremos si el valor imputado es correcto. Por ejemplo, si el valor imputado es mayor que 20 pero en la columna *Indicar título del anticuerpo* aparece "Negativo", entonces cambiaremos el valor imputado para que sea la media de los valores negativos numéricos previos a la imputación.

Además, la médica durante la reunión hizo hincapié en que los resultados de estas pruebas solo tenían validez si se habían realizado con un kit fiable y este es "Aeskulisa tTg-A de Grifols", por lo que eliminaremos todos aquellos resultados que hayan sido obtenidos con un kit distinto a ese.

Finalmente, vamos a unificar estas dos pruebas en una única columna. Según lo que nos dijo la médica, lo que nos interesa es quedarnos con el valor "Positivo" en caso de que lo haya.

- *Indicar título del anticuerpo (DCG ATG2_1), Indicar título del anticuerpo (DCG ATG2_2).* Estas columnas indican el resultado numérico de la prueba para medir el nivel de anticuerpo ATG: contiene valores entre 0 y 300. Como ya hemos dicho, los valores se refieren a la misma prueba realizada en momentos distintos y lo que haremos es unificar estos valores en una misma columna. Para ello, nos quedaremos con el mayor valor. En estas columnas también eliminaremos los valores obtenidos con un kit distinto al nombrado en la columna anterior.

- *Indicar el kit empleado con el punto de corte entre paréntesis.* Estas columnas indican el kit empleado para llevar a cabo la medición de los anticuerpos ATG. Como ya hemos dicho, nos va a permitir desechar los títulos de anticuerpos que se han obtenido utilizando otras pruebas y que por este motivo no resultan fiables. Después de utilizar esta columna para descartar los valores no fiables, eliminaremos esta columna de nuestro dataset.

- *DCG EMA.* Esta columna contiene los resultados a la prueba que mide los anticuerpos Anti-endomisio (EMA) realizadas cuando el paciente estaba siguiendo una dieta con gluten (DCG). Es importante saber que esta técnica puede ser considerablemente cara. Por ello, se suele llevar a cabo solo cuando el paciente tiene un resultado positivo en la prueba ATG2 pero quizá con niveles un poco bajos (se utiliza para confirmar resultados). A su vez, esta prueba tiene una especificidad muy alta, cerca del 100 %, es decir, si un paciente es positivo en esta prueba entonces será celiaco casi con toda seguridad.

  En esta columna tenemos valores "No hechos", "Positivo IgA", "Positivo IgG", "Negativo IgA" y "Negativo IgG". Reduciremos estos valores a "Positivo" y "Negativo" puesto que el "IgA" y el "IgG" solo indican dos variantes de la misma prueba.

- *DCG A-PDG_ 1, DCG A-PDG_ 2, Indique el título del anticuerpo (A-PDG_ 1), Indique el título del anticuerpo (A-PDG_ 2), Indicar el kit empleado con el punto de corte entre paréntesis.* Estas columnas se refieren a las pruebas que miden los valores del anticuerpo PDG cuando el paciente está siguiendo una dieta con gluten (DCG). PDG es un anticuerpo menos específico para el diagnóstico de celiaquía. Tanto es así que hay muchos pacientes que dan positivo en el análisis de este anticuerpo y en cambio no son celiacos, lo que se conoce como falsos positivos. Por esta razón, la médica considera que no es muy importante, aunque nosotros exploraremos estos datos pues la medición de este anticuerpo se utiliza como prueba diagnóstica en la mayoría de hospitales.

  Para el formateo de estas columnas haremos algo muy similar a lo que hicimos para las columnas *ATG2*: la única diferencia es que en este caso el punto de corte es 25 y el kit fiable es "Euroimmun".

- *DSG ATG2_ 1, DSG ATG2_ 2; Indicar titulo del anticuerpo (DSG ATG2_ 1), Indicar título del anticuerpo (DSG ATG2_ 2); Indicar el kit empleado con el punto de corte entre paréntesis.* Estas columnas reflejan lo mismo que las columnas *DCG ATG2* vistas anteriormente pero con la diferencia de que estas pruebas se realizaron cuando el paciente estaba siguiendo una dieta sin gluten. A diferencia de las columnas anteriores, cuando hay 2 valores disponibles, porque la prueba se repitió, seleccionamos el más favorable para el paciente, es decir, el valor "Negativo" en caso de las columnas categóricas y el valor más bajo en caso de las columnas numéricas.

- *DSG A-PDG_ 1, Índique el título del anticuerpo (A-PDG_ 1), A-PDG kit.* Estas columnas reflejan lo mismo que las columnas *DCG A-PDG* vistas anteriormente pero con la diferencia de que estas pruebas se realizaron cuando el paciente estaba siguiendo una dieta con gluten. Al igual que en el caso anterior, cuando hay 2 valores disponibles, seleccionamos el más favorable para el paciente.

- *Fecha DCG Biopsia1, Fecha DCG Biopsia2, FECHA LIEs DCG_ 1, FECHA LIEs DCG_ 2, Fecha DSG biopsia1, Fecha DSG biopsia2, FECHA LIEs DSG_ 1, FECHA LIEs DSG_ 2, FECHA LIEs DSG_ 3.* Estas columnas indican la fecha en la que se hicieron las biopsias. Una biopsia es una prueba cuyo resultado refleja el grado de lesión intestinal. Hay biopsias que se hicieron cuando el paciente estaba siguiendo una dieta con gluten (DCG) y otras cuando el paciente seguía una dieta sin gluten (DSG). Aquellas en las que aparece LIEs están asociadas con biopsias que se hicieron específicamente para dicha prueba. Los resultados de estas pruebas están recogidos en las siguientes columnas. Para reflejar dichos resultados se utiliza la escala de Marsh, donde el valor que refleja un intestino sano es "M0" y va creciendo "M1", "M2", "M3a", "M3b" y "M3c", reflejando cada vez un nivel de lesión más alto del intestino.

  La médica nos indicó que los resultados de las biopsias que se realizan cuando el paciente lleva menos de 6 meses bajo una dieta DSG no son fiables. Por este motivo, lo que haremos es eliminar los resultados de las biopsias DSG para los que hayan transcurrido menos de 6 meses desde que se realizó la última biopsia DCG.

- *DCG Biopsia-AP1, DCG Biopsia-AP2, AP Biopsia DCG LIEs_1, AP en Biopsia DCG LIEs_2.* Estas columnas reflejan los resultados de biopsias realizadas cuando el paciente estaba siguiendo una dieta con gluten (DCG). Lo que haremos con estas columnas es un preprocesado previo para que todas tengan valores de la forma "M0", "M1", ...y luego uniremos todos estos valores en una única columna denominada *Biopsia DCG*, quedándonos con el valor que refleje mayor daño intestinal.

- *DSG Biopsia AP1, DSG Biopsia AP2, AP Biopsia DSG LIEs_1, AP en Biopsia DSG LIEs_2, AP en Biopsia DSG LIEs_3.* Estas columnas reflejan los resultados de las biopsias realizadas cuando el paciente estaba siguiendo una DSG. Lo que haremos con estas columnas es, después de preprocesarlas, combinarlas en una única columna, denominada *Biopsia DSG*, y quedarnos con el valor que refleje menor daño intestinal.

- *Helicobacter pylori en el momento de la biopsia.* Esta columna refleja si el paciente tenía en su intestino la bacteria Helicobacter pylori cuando se le realizó la biopsia. Esta columna contiene valores "Sí"/"No" y, por lo tanto, no tendremos que llevar a cabo ninguna transformación.

- *LIEs DCG %GD_1, LIEs DCG %iNK_1, LIEs DCG %GD_2, LIEs DCG % iNK_2.* La prueba LIEs (*Linfograma de linfocitos intraepiteliales*) se comenzó a utilizar en el diagnóstico de la enfermedad celiaca muy recientemente (en el año 2018) pero es una prueba que está dando muy buenos resultados a la hora de diagnosticar la enfermedad celiaca, especialmente en pacientes atípicos [8]. La prueba consiste en evaluar los parámetros %GD y %iNK. Si se cumple que %GD es mayor o igual que 10 y que %iNK es menor que 10, entonces el paciente es "Compatible con la enfermedad celiaca". Si solo se cumple que %GD es mayor o igual que 10 entonces el paciente se califica como "Compatible con EC en DSG". La explicación es que cuando un paciente está siguiendo una dieta sin gluten, el %iNK aumenta por encima de 10. Los valores numéricos de los parámetros %GD y %iNK están recogidos en estas columnas, mientras que la valoración final está en las columnas siguientes. Como estos resultados numéricos se refieren a la misma prueba realizada en momentos distintos, lo que haremos es unir estos valores en dos columnas, una para %GD y otra para %iNK. De manera que si existe algún par de valores que sea compatible con la enfermedad celiaca entonces nos quedaremos con ese par. En caso que esto no exista, pero sí haya un par de valores que cumplan los requisitos para clasificar al paciente como "EC en DSG" nos quedaremos con ellos. Y por último, si no existe ningún par que cumpla los dos casos anteriores tomaremos aquella prueba que tenga los valores de %GD mayores.

- *Valoración DCG LIEs1, Valoración LIEs2.* Estas columnas contienen los resultados categóricos de las pruebas LIEs. Por tanto los valores de esta columna son "Compatible con EC activa", "Compatible con EC en DSG" y "No compatible con EC". Estas columnas tienen más valores vacíos que las columnas numéricas debido a que cuando los valores numéricos están cerca del límite, la médica prefiere no emitir un diagnóstico. En estos casos rellenaremos los huecos con el valor categórico que corresponda

en función de su valor numérico. Además, combinaremos estas dos columnas en una, priorizando en este orden las valoraciones: “Compatible con EC activa”, “Compatible con EC en DSG” , “No compatible con EC”

- *LIEs DSG %GD_1 , LIEs DSG %iNK_1, LIEs DSG %GD_2, LIEs DSG % iNK_2, LIEs DSG %GD_3, LIEs DSG %iNK_3.* Estas columnas se refieren a la prueba LIEs que se llevó a cabo cuando los pacientes estaban siguiendo una DSG. El tratamiento será idéntico al de las columnas *LIEs DCG*, con la diferencia de que al unir los valores de dos columnas, si existe algún par de valores que sea "No compatible con EC"nos quedaremos con ellos. Si esto no ocurre y existe un par de valores “Compatible con EC en DSG” nos quedaremos con ellos y en cualquier otro caso nos quedaremos con valores “Compatibles con la enfermedad celíaca”.

- *Valoración DSG LIEs1, Valoración DSG LIEs2, Valoración DSG LIEs3.* Estas columnas se refieren a la valoración de las pruebas LIEs cuando el paciente estaba siguiendo una dieta DSG. El tratamiento será idéntico al de las columnas *Valoración DCG LIEs*, consistiendo la única diferencia en que ahora uniremos estas columnas en una sola, priorizando en este orden las valoraciones: “No compatible con EC”, “Compatible con EC en DSG”, “Compatible con EC activa”.

## 2.2. Tratamiento de los datos categóricos

Muchos de los algoritmos de Machine Learning que utilizaremos trabajan únicamente con datos numéricos (de hecho solo el $K$-Modes y el $K$-Prototypes trabajan con datos categóricos). Por ello es necesario que transformemos las columnas categóricas en columnas numéricas. Después de la limpieza y formateo inicial, nuestro dataset tiene las siguientes columnas categóricas: *Indique país de origen o en su defecto la información disponible, Sexo, Edad diagnóstico, HLA: grupos de riesgo, DCG EMA, Helicobacter pylori en el momento de la biopsia, Fecha nacimiento, DCG_ATG2, DCG A-PDG, DSG ATG2, DSG A-PDG, Valoracion LIEs DCG, Valoracion LIEs DSG, Biopsia DCG y Biopsia DSG.*

Por tanto, tenemos que transformar las columnas categóricas a numéricas y para ello existen muchísimos métodos. Por ejemplo, en la librería sklearn hay 17 maneras distintas de hacerlo [18]. Los dos más utilizados son Ordinal encoder [1] y One hot encoder [2]. El *ordinal encoder* asigna a cada categoría un número entero, de forma que cuando aplicamos nuestros algoritmos de clustering, los valores numéricos asignados tienen su valor natural, es decir el 1 está más cerca del 2 que del 4. El *one hot encoder* crea una columna binaria por cada una de las categorías. Esta columna tiene un 1 para cada una de las filas que tiene ese valor categórico y un 0 para las filas que tienen otro valor.

[1] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OrdinalEncoder.html

[2] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html

Puesto que en las columnas categóricas tenemos valores incompletos, no podemos encontrar ninguna ordenación que tenga sentido, dado que ¿cuál es el valor más cercano a un valor desconocido? Lo que haremos es aplicar one hot encoder a todas nuestras columnas categóricas. Para evitar tener que rellenar los datos incompletos con el valor "Desconocido", generar las columnas binarias y luego eliminar estas columnas, lo que haremos es utiliza la función get_dummies [3] de la librería pandas que es equivalente a One hot encoder y puede trabajar con columnas con datos incompletos.

## 2.3. Datos numéricos incompletos

El mayor problema de los datos de los que disponemos es la gran cantidad de valores vacíos. Estos valores vacíos aparecen por dos motivos principales: o bien al paciente aún no se le han realizado todas las pruebas pertinentes o bien el paciente proviene de otro hospital y por ello no se dispone de todo su historial. Los algoritmos de clustering que utilizaremos son, en su mayoría, los implementados en la librería sklearn y no admiten datasets con datos incompletos (de hecho el único algoritmo capaz de trabajar con datos incompletos es $K$-POD). Por este motivo es necesario confrontar este problema. Históricamente se han seguido dos enfoques principales: la *eliminación de datos* y la *imputación*. Explicaremos a continuación cada uno de estos y además, en la sección 3.1.4 explicaremos un algoritmo de clustering especialmente diseñado para el tratamiento de datasets con datos vacíos.

### 2.3.1. Eliminación de datos

Este método es el más simple de todos y uno de los más utilizados. Nuestro dataset, después del formateo y el tratamiento de datos categóricos, tiene en total 66852 celdas. Estas celdas se encuentran repartidas en 620 filas y 108 columnas. Del total de celdas, 3260 tienen un valor nulo, esto es, tenemos un 4,8 % de datos incompletos.

Analizaremos tres enfoques distintos para la eliminación de datos [6; 23] y veremos si obtenemos algún dataset que sea lo suficientemente bueno como para utilizarlo en la aplicación de los algoritmos seleccionados.

#### Eliminación de columnas con datos incompletos

Esta primera técnica es la peor de todas dado que todas las columnas que tenemos en nuestro dataset son útiles. Ello es así porque provienen del formateo previo que se hizo siguiendo las recomendaciones de la médica. Por tanto, perder un (gran) número de columnas supone una pérdida relevante.

Después de eliminar las columnas con datos incompletos, nuestro dataset tiene 100 columnas lo que en número no parece que esté nada mal (partíamos de 108 columnas). Sin

[3] https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.get_dummies.html

embargo, un estudio detenido de las columnas eliminadas[4] muestra que se trata de columnas relativas a los resultados numéricos de las pruebas diagnósticas cuya información, a pesar de estar directamente relacionada con las columnas categóricas, la médica considera fundamental.

Por tanto, este enfoque es incompatible con nuestros datos y queda descartado.

#### Eliminación de filas con datos incompletos

Esta técnica consiste en eliminar todas aquellas filas que tienen algún valor nulo. En un dataset grande puede que esta técnica no tenga efectos demasiado graves, pero en nuestro caso contamos con tan solo 600 pacientes y nos supondría perder información muy valiosa. De hecho, después de eliminar las filas con datos incompletos nuestro dataset tiene una única fila, por lo que este enfoque queda completamente descartado.

#### Eliminación según porcentajes

Esta técnica consiste en calcular, para todas las filas y todas las columnas, el porcentaje de datos incompletos. A continuación, se eliminan las filas o columnas que tengan el mayor porcentaje de valores nulos. Este proceso se itera hasta que no quede ningún elemento nulo en nuestro dataset.

Después de llevar a cabo este proceso nos quedamos con 100 columnas y 619 filas. Las columnas que se han eliminado son las mismas que en el primer enfoque y, por tanto, esta alternativa tampoco es viable.

#### Conclusiones sobre la eliminación

A pesar de ser un enfoque muy utilizado, por las características de nuestro dataset resulta inviable utilizar la eliminación puesto que perderíamos columnas que resultan imprescindibles para nuestro estudio.

### 2.3.2. Imputación

La imputación consiste en completar las celdas vacías con algún valor que tenga sentido. Una primera aproximación *naïve* consiste en rellenar todos los valores vacíos con un valor numérico que indique que no hay valor. Desgraciadamente, esta solución no es adecuada dado que si pusiéramos, por ejemplo, el valor -1 entonces el algoritmo de clustering entendería que este valor está más cerca de 0 que de 300, cuando lo que en realidad querríamos es que estuviera a la misma distancia de todos ellos.

Las técnicas de imputación [15] que vamos a revisar se pueden utilizar para imputar tanto valores numéricos como valores categóricos. Nótese que en nuestro caso concreto, para estos últimos no utilizaremos esta opción dado que, como ya hemos explicado, hemos generado

[4]Las columnas eliminadas son: *DCG_ATG2_ VALUE, DCG A-PDG_ VALUE, LIEs DCG %iNK, DSG ATG2 VALUE, LIEs DSG %GD, LIEs DSG %iNK, LIEs DCG %GD y DSG A-PDG VALUE.*

columnas binarias para representarlos en el caso de los datasets numéricos o hemos añadido una nueva categoría "Desconocido" en el caso de los datasets que conservan las columnas categóricas. Estudiaremos las distintas técnicas y veremos cual es la más interesante para nuestro dataset.

### Imputación univariada

Este tipo de imputación consiste en asignar valores a los elementos vacíos de una columna basándose únicamente en los valores no nulos de esa columna. Se podría, por ejemplo, imputar la media, la moda, o la mediana de los valores de esa columna. Esta imputación no tiene sentido en nuestro dataset ya que al estar tratando con los resultados de pruebas diagnósticas, estos no dependen de ninguna forma de los resultados del resto de pacientes.

### Imputación multivariada

En este tipo de imputación cada una de las columnas que tienen algún dato incompleto se expresa como función de los valores de las otras columnas. Después, se utiliza esta función para calcular los valores que faltan. Puede parecer que está imputación sí tiene sentido en nuestro dataset ya que existe cierta relación entre los resultados de las distintas pruebas diagnósticas o entre los distintos indicadores. Por ejemplo, si un paciente tiene una genética que le hace pertenecer a un grupo de riesgo, por ejemplo DQ2.5 doble dosis, entonces es más probable que tenga EC y por lo tanto que los resultados de la prueba ATG2 tengan un valor mayor que 20.

Hemos ejecutado la implementación de sklearn de este imputador IterativeImputer [5] sobre nuestro dataset. Concretamente, este imputador funciona de la siguiente manera. Para cada columna $y$ que tiene un valor vacío, generamos la matriz $(X, y)$ donde $X$ contiene el resto de columnas y se entrena a un regresor que debe predecir los valores desconocidos de $y$. Este proceso se repite tantas veces como columnas con valores vacíos haya y hasta que se alcance un número máximo de iteraciones. En nuestro caso, cuando hemos utilizado este imputador sobre nuestros datos hemos tenido problemas de convergencia aún utilizando un número muy alto de iteraciones (en concreto 100) por lo que hemos concluido esta imputación es incompatible con nuestro dataset.

### Imputación por los vecinos más cercanos

En este tipo de imputación, los valores incompletos se rellenan utilizando los valores de los vecinos más cercanos. Dado un paciente $x$ que tenga un valor incompleto en la columna $c$ encontramos los $k$ pacientes que estén a menor distancia del actual y que tengan un valor no nulo en $c$ y asignamos la media de esos valores a la columna $c$ del paciente $x$. Para calcular la *distancia* entre pacientes, se tienen en cuenta las columnas que no están incompletas para ninguno de los dos pacientes.

[5] https://scikit-learn.org/stable/modules/generated/sklearn.impute.IterativeImputer.html#sklearn.impute.IterativeImputer

Esta imputación sí tiene sentido en nuestro dataset porque cabe esperar que pacientes con una clínica parecida tengan unos datos parecidos. Hemos utilizado la implementación disponible en sklearn KNNImputer[6] sobre nuestro dataset y no hemos tenido ningún problema. Hemos elegido como parámetro inicial $k = 4$ (es decir, tendremos en cuenta para imputar el valor a los 4 pacientes más cercanos al paciente en cuestión), aunque este parámetro lo podemos cambiar posteriormente.

## 2.4. Escalado de los datos

Si la unificación de la escala de los datos numéricos es una técnica básica en *machine learning*, pues mejora los tiempos de convergencia, es aún más importante cuando aplicamos algoritmos de clustering, pues lo que estamos haciendo es calcular la distancia entre elementos de nuestro dataset y, para ello, estamos utilizando los valores de todas las columnas. Si las escalas de las columnas (numéricas) son distintas, entonces nuestro algoritmo solo tendrá en cuenta las columnas que tienen valores mayores. Por ejemplo, en nuestro dataset tenemos muchas columnas binarias mientras que otras, como *DCG_ATG2_VALUE*, tienen valores del 0 al 300. En caso de que no unificáramos la escala de las variables numéricas, nuestros algoritmos no tendrían prácticamente en cuenta los datos de las columnas binarias.

Existen muchas formas de escalar los datos [11]. La aplicación de uno u otro método depende de la distribución que sigan los datos, de la presencia de valores atípicos y de la relación (lineal o no) entre las columnas. Explicaremos brevemente aquí algunas de las técnicas (aquellas que están implementadas en sklearn en el módulo de preprocesado[7]) y diremos cuál hemos elegido para cada una de nuestras columnas en el dataset. Específicamente, después del formateo y la limpieza previa, las columnas numéricas que necesitan normalización son: *DCG_ATG2_VALUE, DCG A-PDG_VALUE, DSG ATG2 VALUE, DSG A-PDG VALUE, LIEs DCG %GD, LIEs DCG %iNK, LIEs DSG %GD* y *LIEs DSG %iNK*. En la figura 2.1 podemos ver gráficamente la distribución de nuestras variables numéricas.

### MinMax Scaler

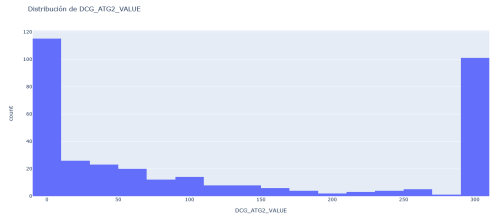
Este método consiste en escalar los valores para que estén dentro de un rango que normalmente es $[0, 1]$. Para ello, para todo $x$ de la columna $c$ se aplica la transformación

$$x_{escalado} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

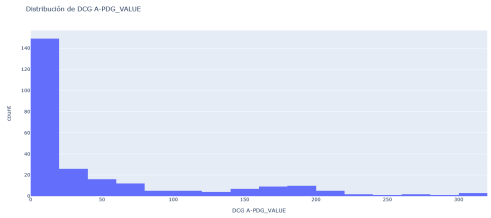
donde $x_{min}$ es el mínimo valor de la columna $c$ y $x_{max}$ el máximo.

[6] https://scikit-learn.org/stable/modules/generated/sklearn.impute.KNNImputer.html#sklearn.impute.KNNImputer

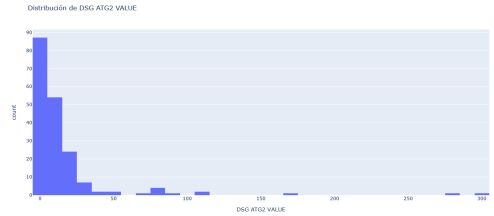
[7] https://scikit-learn.org/stable/modules/preprocessing.html#preprocessing



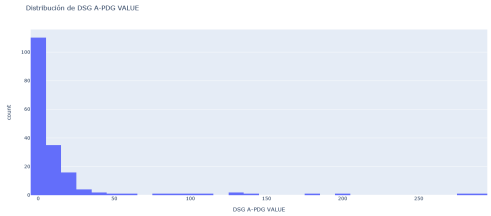

(a) Distribución de DCG_ATG2_VALUE



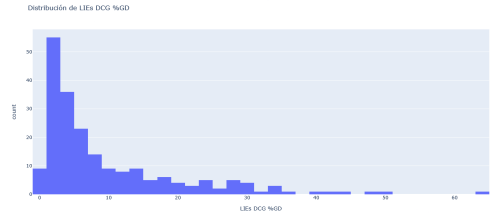

(b) Distribución de DCG A-PDG_VALUE



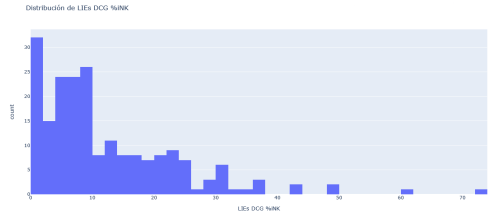

(c) Distribución de DSG ATG2 VALUE



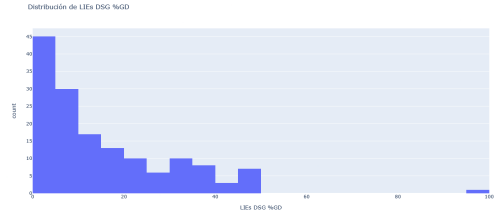

(d) Distribución de DSG A-PDG VALUE



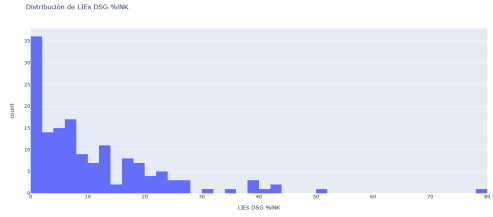

(e) Distribución de LIEs DCG %GD




(f) Distribución de LIEs DCG %iNK




(g) Distribución de LIEs DSG %GD




(h) Distribución de LIEs DSG %iNK

Figura 2.1: Distribución de las variables numéricas

### MaxAbs Scaler

Este método es muy similar al anterior: escala los valores para que estos estén en el rango $[-1, 1]$. Para ello, para todo $x$ de la columna $c$ se aplica la transformación

$$x_{escalado} = \frac{x}{|x_{max}|}$$

En nuestro caso solo tenemos valores positivos por lo que este método, al igual que el anterior, escala los valores en el rango $[0, 1]$ y se comportan de una manera similar.

### Robust Scaler

Los métodos anteriores sufren mucho con los valores atípicos [14] pues están muy influenciados por los valores máximos y mínimos. Esto supone un problema para nosotros, pues tenemos columnas en nuestro dataset, como *LIES DSG %GD*, para las cuales un único individuo toma el valor máximo (en este caso 95,6 %) y el resto toma valores entre 0 y 49. Por tanto, aplicar un reescalado a todos los valores dará lugar a que estén entre 0 y 0,51, excepto un único valor que será igual a 1. Existen otras variables, por ejemplo *DSG ATG2 VALUE*, que también tienen valores atípicos (véase la figura 2.1).

Para evitar los problemas introducidos por la existencia de valores atípicos podemos utilizar *Robust Scaler*, un método que en lugar de utilizar el máximo y el mínimo de las columnas para escalar utiliza los cuartiles. Formalmente, para todo $x$ de la columna $c$ se aplica la transformación

$$x_{escalado} = \frac{x - Q_1}{Q_3 - Q_1}$$

El problema es que, en este caso, nuestros datos no tomarán valores pequeños necesariamente. De hecho, hemos aplicado este método sobre nuestros datos y detectamos graves problemas dado que en la columna *DSG ATG2 VALUE* seguía habiendo valores considerablemente grandes: tomaban el valor 25. Por tanto, si aplicamos un algoritmo de clustering basado en distancias, por ejemplo K-Means, nuestro algoritmo solo tendría en cuenta la variable *DSG ATG2 VALUE*.

### Standard Scaler

Este método escala los valores de manera que tengan media 0 y varianza 1. Para conseguir este objetivo, para cada valor $x$ se aplica la transformación

$$x_{escalado} = \frac{x - media}{varianza}$$

Desgraciadamente, este método asume que los datos tienen una distribución normal, pero es obvio que nuestros datos no siguen dicha distribución.

Por lo explicado anteriormente, ninguno de los métodos que hemos presentado nos sirven para reescalar nuestro dataset. Por ello, nos vimos abocados a buscar algún método que cambiara la distribución de los datos, con el objetivo de eliminar o disminuir los valores atípicos, y luego reescalar.

### Quantile Transformer Scaler

Este método transforma la distribución de los datos para que sigan una distribución normal. Esencialmente, la aplicación de este método consiste en decidir el número de intervalos en los que se van a dividir los datos, definir los intervalos de manera que en cada uno de ellos haya el mismo número de elementos, se sustituye cada elemento por el intervalo al que pertenece y finalmente se escalan los datos normalmente para que estén en el intervalo

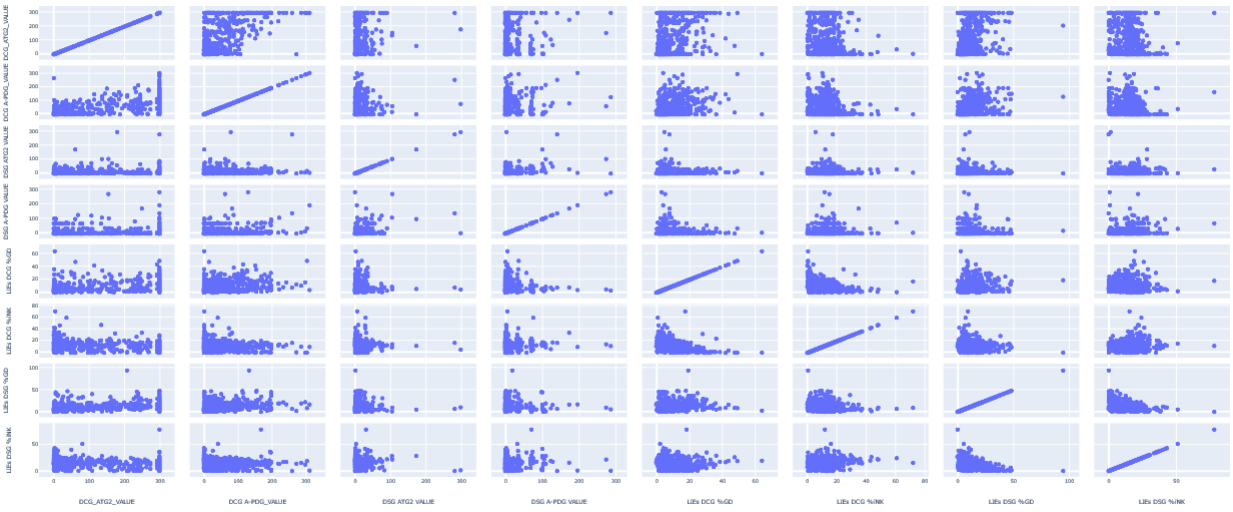


Figura 2.2: Relación entre las variables numéricas

$[0, 1]$. Hay que tener en cuenta que esta transformación no es lineal y, por lo tanto, puede distorsionar las correlaciones lineales entre variables [11], aunque esto no supone ningún problema porque como podemos ver en la figura 2.2, las variables numéricas no tienen relaciones lineales entre ellas[8] Crearemos solo 6 intervalos o cuantiles porque tenemos del orden de 600 pacientes [5].

## 2.5. Datasets generados

En primer lugar hay que tener en cuenta que, a pesar de que la mayoría de nuestros algoritmos trabajan con datasets numéricos con datos completos existen algunos como $K$-Prototypes, $K$-Modes o $K$-POD que utilizan otro tipos de datasets por lo que generaremos los siguientes tipos:

- *Dataset numérico.* En general los dataset numéricos son aquellos en los que las columnas categóricas se han transformados en columnas binarias. Además no contienen datos incompletos por los que se ha llevado a cabo la imputación de los datos y el posterior escalado de todas las columnas.

- *Dataset categórico.* Este dataset solo se va a utilizar con el algoritmo K-Modes e incluye todas las columnas categóricas y las columnas relativas a grado de parentesco, las generadas por enfermedad inmunológica, síntomas y signos (estas columnas aunque no son categóricas, son binarias y se pueden tratar como tal). Tiene sentido desprendernos

[8] Nótese que las gráficas que aparecen en la *diagonal* de la figura 2.2, que muestran una correlación lineal obvia, corresponden a relaciones entre una variable columna y si misma.

de las columnas numéricas ya que su información ya está recogida en las columnas categóricas. La única transformación que se lleva a cabo después del formateo inicial es rellenar con "Desconocido" las celdas vacías. Es decir, puesto que no trabajamos con las columnas numéricas no tenemos que hacer los pasos de transformación, imputación y escalado.

- *Dataset mixto.* Este dataset solo se va a utilizar con el algoritmo K-Prototypes e incluye todas la columnas categóricas y todas las columnas numéricas. La única transformación que se lleva a cabo en las columnas categóricas consiste en rellenar con "Desconocido" las celdas vacías (es decir, no se transforman en columnas binarias). A continuación se hace la imputación y el escalado de los datos numéricos.

- *Dataset incompleto.* Este dataset solo lo vamos a utilizar con el algoritmo K-POD. Es idéntico al dataset numérico pero no se ha hecho la imputación de los datos.

A medida que íbamos trabajando en el formateo de los datos nos íbamos dando cuenta de que tenía sentido construir distintos datasets y ver qué tal se comportan nuestros modelos sobre ellos. Por lo que construimos datasets que contenían subconjuntos de las columnas del dataset inicial. Detallaremos los datasets generados para el tipo numérico pero hay que tener en cuenta que esos mismos se generan para los datasets categórico, mixto e incompleto.

- *Dataset numérico 1: completo.* Este dataset incluye todas las columnas relevantes. A las columnas categóricas se le han hecho las transformaciones pertinentes, se ha llevado a cabo la imputación y el escalado de los datos.

- *Dataset numérico 2: resumido.* Este dataset incluye todas las columnas numéricas pero no se han incluido todas las columnas categóricas. De este último tipo se incluyen únicamente las no redundantes (es decir, las que no tienen una columna numérica asociada) que son: *Indique país de origen o en su defecto la información disponible, Sexo, Edad diagnóstico, HLA: grupos de riesgo, DCG EMA, DSG EMA, Helicobacter pylori en el momento de la biopsia, Fecha nacimiento, Biopsia DCG y Biopsia DSG*. Sobre este dataset, se han hecho las transformaciones pertinentes a las columnas categóricas y se ha hecho la imputación y el escalado.

- *Dataset numérico 3: sin país, sexo, edad diagnóstico ni grado de parentesco.* Este dataset incluye las mismas columnas que el dataset numérico pero eliminando las columnas *Indique país de origen o en su defecto la información disponible, Sexo, Edad diagnóstico, Grado de parentesco y Grado de parentesco (si hay más de 1)*. Hemos eliminado estas columnas porque tienen que ver con la genética que ya viene recogida en posteriores columnas. Para el resto de datos, se siguen las mismas transformaciones que para el dataset numérico completo.

- *Dataset numérico 4: sin país, sexo, edad diagnóstico ni grado de parentesco resumido.* Es el mismo dataset que el anterior pero eliminando las columnas mencionadas en el dataset numérico resumido.

- *Dataset numérico 5: sin signos ni síntomas.* Es un dataset generado a partir del datset numérico completo pero eliminando todas las columnas de *Signos y Síntomas*.

- *Dataset numérico 6: sin signos ni síntomas acortado.* Es el mismo dataset que el anterior pero eliminando las columnas vistas en el dataset numérico acortado.

- *Dataset numérico 7: mezcla entre 3 y 5.* Es la combinación entre el dataset numérico 3 y 5. Es decir, partiendo del dataset numérico completo, se eliminan las columnas que se borran en los 2 datasets mencionados.

- *Dataset numérico 8: dataset 7 acortado.* Tomando de partida el dataset numérico 7, se eliminan las columnas que se eliminan en el dataset acortado 2.

- *Dataset con las columnas relevantes en el diagnóstico de la celiaquía.* Después de acudir a las conferencias del *VII Congreso Nacional de la Sociedad Española de Enfermedad Celíaca* adquirimos un mayor conocimiento sobre la enfermedad celíaca y por ello generamos un dataset más que contuviera únicamente las columnas más relevantes en el diagnóstico de la celiaquía. Estas columnas son las relacionadas con la genética, es decir *HLA: grupos de riesgo* y *1º grado*, las columnas de serología cuando el paciente sigue dieta DCG, es decir *DCG EMA, Valoración DCG_ATG2, Valoración DCG A-PDG* y *Valoración LIEs DCG*. También tuvimos en cuenta los síntomas y signos más comunes que son *Diarrea crónica, Estreñimiento, Distensión abdominal, Dispepsia, Malabsorción* y *Anemia ferropénica o ferropenia* y los resultados de la biopsia en DCG, es decir la columna *Biopsia DCG*. Además, incluimos las columnas *Biopsia DSG* y *Valoración LIEs DSG* porque ambas contienen valores característicos en los casos de enfermos celíacos que empiezan una dieta sin gluten, en el caso de la Biopsia vuelve a valores normales y la prueba *LIEs* tiene un comportamiento característico para este tipo de pacientes.

- *Dataset con atributos seleccionados mediante CFS(Correlation based Feature Selection).* Utilizando el dataset con todas las columnas y solo con aquellos pacientes que tuvieran diagnósticos distintos de *Paciente perdido, Sin diagnóstico y Aún en estudio* ejecutamos esta técnica de selección de variables que seleccionó las columnas: *LIEs DSG %GD, Biopsia DCG_M0, Valoracion LIEs DCG_No compatible con EC, DCG_ATG2_Negativo, DCG_ATG2_VALUE, Valoracion LIEs DCG_Compatible con EC activa, DSG ATG2_Negativo, LIEs DCG %GD, DCG A-PDG_Negativo, DCG _ATG2_Positivo, Biopsia DCG_M3b*.

- *Dataset con atributos seleccionados mediante FCBF(Fast Correlation-Base Filter).* Como en el caso anterior utilizamos esta técnica de selección de variables sobre el dataset con todas las columnas al que se le eliminó a los pacientes con diagnósticos *Paciente perdido, Sin diagnóstico y Aún en estudio*. Esta técnica selecciónó las columnas: *LIEs DSG %GD, Biopsia DCG_M0, Valoracion LIEs DCG_No compatible con EC, DCG_ATG2_Negativo, Malabsorción, Biopsia DSG_M3b, Esclerosis múltiple*.

# Capítulo 3

# Base teórica de los algoritmos de clústering

En este capítulo presentaremos los diferentes algoritmos de clústering y sus principales características. Existen infinidad de algoritmos y métodos para realizar la agrupación de datos que se pueden clasificar en dos familias perfectamente diferenciadas. La primera de ellas engloba a los algoritmos de *clústering difuso* o *suave* (*fuzzy algorithms*). Esta familia de algoritmos se caracteriza por indicar la probabilidad que tiene un dato de pertenecer a un cierto clúster. Dicha probabilidad está en un intervalo entre $[0, 1]$ donde 0 indica que es imposible que un dato pertenezca a un cierto clúster y 1 que pertenece con total seguridad. Por lo tanto, un dato puede tener probabilidad mayor que cero de pertenecer a varios clústeres. Esta familia de algoritmos no ha sido utilizada en este trabajo para el análisis de los datos. Por otra parte tenemos la familia de algoritmos de *clústering fuerte*. Todos los métodos que la forman tiene en común que cada dato acaba clasificado en un único clúster. Es decir, los datos únicamente pertenecen a un grupo. Además, es posible distinguir varias subfamilias dentro de la familia de algoritmos de clústering duro. Las diferencias entre ellas residen en la forma que tienen de crear los clústeres, así como los parámetros iniciales que reciben. Cada una de estas subfamilias contienen a su vez muchos algoritmos con pequeñas diferencias entre sí. A lo largo de este trabajo se han utilizado diferentes subfamilias y diferentes métodos de las mismas familias para tratar de encontrar un algoritmo que se adapte mejor a nuestro conjunto de datos inicial y con el que poder obtener mejores resultados. Durante el resto de este capítulo detallaremos las subfamilias y algoritmos que hemos utilizado para nuestro análisis de datos.

## 3.1. Algoritmos de partición

Esta familia de algoritmos engloba una gran variedad de métodos, aunque en el fondo todos siguen el mismo proceso: dividir el conjunto de datos iniciales en diferentes grupos a partir de ciertos puntos. Sin embargo, la forma en la que llevan a cabo este proceso cambia mucho de unos a otros y existen diferencias importantes entre ellos. Comparado con otras

familias de algoritmos, estos métodos son muy eficientes para agrupar grandes conjuntos de datos con multitud de campos porque suelen requerir poco tiempo de ejecución y poco espacio para guardar los datos. Normalmente, tienen sus propias funciones objetivo que indican cuando una partición es buena, con la finalidad de encontrar una partición que minimice el valor de esta función objetivo.

La técnica general de este tipo de algoritmos suele consistir en emplear un punto *importante* en los diferentes clústeres, llamado centroide, y a partir de dicho punto comenzar la partición. De forma iterativa, se va asignando a cada punto del conjunto de datos el clúster cuyo centroide está más cerca de dicho punto para así tratar de minimizar la función objetivo. Las pasos posteriores que sirven para optimizar la función objetivo dependen de cada algoritmo. Pasamos a continuación a describir las variantes más relevantes dentro de esta subfamilia de algoritmos.

### 3.1.1. *K*-means

Este algoritmo es el más conocido y utilizado hoy en día [13; 17]. Es posible que este *éxito* se deba a que la idea detrás del mismo es extremadamente sencilla.

#### Base teórica

El algoritmo $K$-means busca agrupar un conjunto de $M$ elementos que están en un espacio $n$ dimensional en $K$ clústeres. Este conjunto de elementos se define como $S = \{x_1, ..., x_M\}$ donde $x_i = (X_{i,r})_{r=1}^n$ para todo $i \in \{1, ..., M\}$. Además, necesitamos una medida de disimilitud entre los elementos de $S$. Dados $x_i, x_j \in S$, donde $x_i = (X_{i,r})_{r=1}^n$ y $x_j = (X_{j,r})_{r=1}^n$, la *disimilitud* entre estos dos elementos se define como:

$$d(x_i, x_j) = (\sum_{r=1}^{n} |X_{i,r} - X_{j,r}|^2) \tag{3.1}$$

El objetivo del algoritmo es formar clústeres que contengan elementos lo más parecidos posibles entre sí, mientras que los diferentes clústeres sean lo más distintos posibles. Esto es equivalente a minimizar la suma de las disimilitudes entre los elementos del clúster y maximizar la disimilitud entre los elementos de un clúster y los elementos de los otros. Sea un conjunto de elementos $S = \{x_1, ...x_M\}$ que están distribuidos en $K$ clústeres $C = \{C_q\}_{q=1}^K$ donde cada clúster $C_q$ tiene un centro $c_q \in \mathbb{R}^n$. Debemos minimizar la siguiente función:

$$intraSimilitud = \sum_{C_i \in C} \sum_{x_i, x_j \in C_i} d(x_i, x_j) \tag{3.2}$$

Esto es equivalente a minimizar la función:

$$intraSimilitud_2 = \sum_{C_i \in C} \frac{1}{2|C_i|} \sum_{x_i \in C_i} d(x_i, c_i) \tag{3.3}$$

Por otra parte, debemos maximizar la función:

$$interSimilitud = \sum_{C_i \in C} \sum_{C_j \in C} \sum_{x_i \in C_i} \sum_{x_j \in C_j} d(x_i, x_j) \tag{3.4}$$

Nótese que como la suma de las similitudes entre los elementos de conjunto $S$ permanece constante y esta es igual a $intraSimilitud + interSimilitud$, minimizar la $intraSimilitud$ es equivalente a maximizar la $interSimilitud$. Por lo tanto, para optimizar el valor de estas cantidades podemos realizar los siguientes pasos:

- Para minimizar la similitud intraclúster, debemos minimizar la similitud de cada punto al centro de su clúster. Para ello asignaremos cada elemento al clúster con cuyo centro tenga mayor similitud.
- Utilizamos como el centro del clúster la media de los elementos que forman parte del clúster porque es la cantidad que minimiza la suma de las diferencias al cuadrado.

**Algoritmo**

Partimos de un conjunto de elementos $S = \{x_1, ..., x_M\}$, el número de clústeres $K$ que deseamos construir y un conjunto de centros (medias) $c = \{c_q\}_{q=1}^{K}$ iniciales elegidas entre los elementos del conjunto (por ejemplo, aleatoriamente). El algoritmo $K$-means es un algoritmo iterativo que alterna los siguientes dos pasos hasta que se cumple una condición que suele ser que los centros de una iteración estén muy cercanos a los centros de la iteración siguiente.

1. Paso de asignación. Se asigna cada punto de $S$ al centro que esté más cercano. Estamos construyendo por lo tanto los siguientes clústeres:

   $$C_i^{(t)} = \{x_l \in S | d(x_l, c_i^{(t)}) \leq d(x_l, c_j^{(t)}) \forall j \in \{1, ...K\}\}$$

   Si un punto tiene la misma similitud con dos centros, se asignará al de menor índice.

2. Paso de actualización. Recalculamos los centros de los clústeres (medias) como la media en cada coordenada de los elementos del clúster:

   $$c_i^{(t+1)} = \frac{1}{|C_i^{(t)}|} \sum_{x_j \in C_i^{(t)}} x_j$$

   Notemos que estos elementos no tienen por qué pertenecer al conjunto inicial $S$.

### 3.1.2. *K*-prototypes

Este algoritmo [10] es una variación del $K$-means que nos permite trabajar con datos mixtos, es decir, con un dataset que contiene datos categóricos y datos numéricos.

**Base teórica**

Sea un conjunto de elementos $S = \{x_1, ..., x_M\}$ donde $x_i = (X_{i,r})_{r=1}^n$ para todo $i \in \{1, ..., M\}$. Suponemos, sin pérdida de generalidad, que las componentes $\{1, ..., m_s\}$ de cada elemento se corresponden con datos numéricos y las componentes $\{m_s + 1, ..., n\}$ se corresponden con los datos categóricos. Entonces definimos la medida de *disimilitud* entre dos elementos $x_i = (X_{i,r})_{r=1}^n$ y $x_j = (X_{j,r})_{r=1}^n$ de $S$ como:

$$d(x_i, x_j) = \sum_{r=1}^{m_s} (X_{i,r} - X_{j,r})^2 + \sum_{r=m_s+1}^{n} \delta(X_{i,r}, X_{j,r})$$

donde $\delta(p, q) = 0$ si $p = q$ y $\delta(p, q) = 1$ si $p \neq q$.

Lo que buscamos es que los clústeres contengan elementos que sean lo más parecidos posibles. Esto es equivalente a minimizar la suma de las disimilitudes entre los elementos del clúster y el centro del mismo. Matemáticamente, sea un conjunto de elementos $S = \{x_1, ..., x_M\}$ donde $x_i = (X_{i,r})_{r=1}^n$ para todo $i \in \{1, ..., M\}$ donde las componentes $\{1, ..., m_s\}$ contienen datos numéricos y $\{m_s+1, ..., n\}$ datos categóricos. Estos puntos están distribuidos en $K$ clústeres $C = \{C_q\}_{q=1}^K$, donde cada clúster $C_q$ tiene un centro $c_q = (c_{q,r})_{r=1}^n$. Nuestro objetivo es minimizar la función:

$$E = \sum_{C_i \in C} \sum_{x_i \in C_i} d(x_i, c_i) = \sum_{C_i \in C} \sum_{x_i \in C_i} (\sum_{r=1}^{m_s} (X_{i,r} - c_{i,r})^2 + \sum_{r=m_s+1}^{n} \delta(X_{i,r}, c_{i,r})) = E_{num} + E_{cat}$$

La expresión $E_{cat}$ se puede expresar de forma equivalente utilizando probabilidades. Si denotamos por $CAT_j$ al conjunto que contiene todos los valores categóricos de la columna $j$-ésima, $j \in \{m_s + 1, ..., n\}$, y por $p(cat \in CAT_j | C_l)$ a la probabilidad de que el valor $cat$ aparezca en un elemento del clúster $l$-ésimo, entonces la nueva forma de escribir $E_{cat}$ es:

$$E_{cat} = \sum_{C_i \in C} \sum_{x_i \in C_i} \sum_{r=m_s+1}^{n} n_i (1 - p(c_{i,r} \in CAT_r | C_i))$$

donde $n_i$ es el número de elementos que hay en el clúster $i$-ésimo. Tenemos el siguiente resultado.

**Lema 3.1.1.** *$E_{cat}$ se minimiza cuando para todo clúster $C_i$ se cumple $p(c_{i,j} \in CAT_j | C_i) \geq p(cat \in CAT_j | C_i)$, para todo cat tal que $cat \neq c_{i,j}$. Es decir, cuando la probabilidad de que el valor categórico de los individuos del clúster coincida con el valor del centro (o media) es mayor que para cualquier otro valor.*

Además como $E_{num}$ y $E_{cat}$ son cantidades no negativas, minimizar $E$ es equivalente a minimizar $E_{num}$ y $E_{cat}$. Por lo tanto, $E_{num}$ se minimiza eligiendo los elementos numéricos de los centros como en $K$-means y $E_{cat}$ se minimiza seleccionando los elementos categóricos de los centros del clúster según el Lema 3.1.1.

**Algoritmo**

Partimos de un conjunto de elementos $S = \{x_1, ..., x_M\}$ donde $x_i = (X_{i,r})_{r=1}^n$ para todo $i \in \{1, ..., M\}$, el número de clústeres $K$ que deseamos construir y un conjunto de centros (prototipos) $c = \{c_q\}_{q=1}^K$ iniciales elegidos entre los elementos del conjunto (por ejemplo, aleatoriamente). El algoritmo $K$-prototypes es un algoritmo iterativo que alterna los siguientes dos pasos hasta que ningún elemento cambia de clúster asignado.

1. Paso de asignación. Se asigna cada punto de $S$ al centro que esté más cercano. Por lo tanto, estamos construyendo los clústeres aplicando la misma fórmula que para $K$-means.

2. Paso de actualización. Recalculamos los centros de los clúster (prototipos) como la media en cada coordenada numérica y como el valor categórico con mayor probabilidad (el que más se repita) en cada coordenada categórica de los elementos del clúster:

$$c_{i,r}^{(t+1)} = \frac{1}{|C_i^{(t)}|} \sum_{x_j \in C_i^{(t)}} X_{j,r} \quad \forall r \in \{1, ..., m_s\}$$

$$c_{i,r}^{(t+1)} = \text{máx}\{p(X_{j,r} \in CAT_r | C_i) | x_j \in C_i\} \quad \forall r \in \{m_s + 1, ..., n\}$$

### 3.1.3. *K*-modes

Este algoritmo es otra variación del $K$-means que nos permite trabajar con un dataset que contiene únicamente datos categóricos. Su implementación y justificación coincide con la de $K$-prototypes, consistiendo la principal diferencia en que como no hay columnas numéricas, la definición de la medida de disimilitud, la función a minimizar y el paso de actualización del algoritmo tienen que adaptarse a esta situación:

$$d(x_i, x_j) = \sum_{r=1}^{n} \delta(X_{i,r}, X_{j,r})$$

$$E = \sum_{C_i \in C} \sum_{x_i \in C_i} d(x_i, c_i) = \sum_{r=1}^{n} \delta(X_{i,r}, c_{i,r}))$$

$$c_{i,r}^{(t+1)} = \text{máx}\{p(X_{j,r} \in CAT_r | C_i) : x_j \in C_i\} \quad \forall r \in \{1, ..., n\}$$

### 3.1.4. *K*-POD

Este algoritmo [4] es otra variación del algoritmo $K$-means que fue especialmente diseñado para trabajar con datasets incompletos. Este algoritmo resulta interesante en nuestro ámbito concreto ya que tenemos un gran porcentaje del dataset incompleto. Aunque utilizamos métodos de imputación, nos gustaría estudiar si este algoritmo se comporta mejor que otros algoritmos sobre nuestros datos imputados.

Intuitivamente $K$-POD completa los valores de los campos incompletos de los elementos dándoles el valor del centro del clúster. Una vez el dataset está completo, se aplica el algoritmo $K$-means. Iteramos este proceso hasta que los centroides no cambien.

### Base teórica

Sea un conjunto de elementos $S = \{x_1, ..., x_M\}$ donde $x_i = (X_{i,r})_{r=1}^n$ para todo $i \in \{1, ..., M\}$ que están distribuidos en $K$ clústeres $C = \{C_q\}_{q=1}^K$, donde cada clúster $C_q$ tiene un centro $c_q \in \mathbb{R}^n$. Debemos minimizar:

$$intraSimilitud_2 = \sum_{C_i \in C} \sum_{x_i \in C_i} d(x_i, c_i) = \sum_{C_i \in C} \sum_{x_i \in C_i} \|x_i - c_i\|_2^2$$

Si definimos $X = (X_{j,r})_{j \in \{1...M\}, r \in \{1,...n\}} \in \mathbb{R}^{M \times n}$, $B \in \mathbb{R}^{K \times n}$ la matriz cuyas filas son los centros de los clústeres y $A \in \{0,1\}^{M \times K}$ una matriz binaria que indica la pertenencia de un elemento a un clúster (es decir, $a_{ij} = 1$ si y solo si $x_i \in C_j$), entonces podemos reescribir $intraSimilitud_2$ matricialmente como:

$$\text{mín} \, \|X - AB\|_F^2$$

siendo $\|\cdot\|_F^2$ la norma de Frobenius de una matriz que se define como $||A||_F^2 = \sum_{i,j} a_{ij}^2$.

Si buscamos minimizar la misma expresión pero teniendo en cuenta el dataset incompleto, debemos definir un conjunto de tuplas $\Omega \subset \{1, ..., M\} \times \{1, ..., n\}$ que corresponden con las componentes de los elementos del conjunto que están completas. Ahora definimos la proyección de una matriz sobre este conjunto como:

$$|P_\Omega(Y)|_{ij} = \begin{cases} y_{ij} & \text{si}(i,j) \in \Omega \\ 0 & \text{si}(i,j) \in \Omega^c \end{cases}$$

donde $\Omega^c$ es el conjunto complemetario de $\Omega$. De forma natural, obtenemos que el problema que queremos resolver es:

$$\text{mín} \, \|P_\Omega(X) - P_\Omega(AB)\|_F^2$$

La demostración del siguiente resultado se puede encontrar en [4].

**Lema 3.1.2.** *Sean $A$ y $B$ las matrices definidas anteriormente y sea $(\widetilde{A}, \widetilde{B})$ la salida de aplicar el algoritmo $K$-means a $X = P_\Omega(X) + P_{\Omega^C}(AB)$. Se cumple:*

$$\|P_\Omega(X) - P_\Omega(AB)\|_F^2 \geq \left\| P_\Omega(X) - P_\Omega(\widetilde{A}\widetilde{B}) \right\|_F^2$$

### Algoritmo

Partimos de un conjunto de elementos $S = \{x_1, ..., x_M\}$ donde $x_i = (X_{i,r})_{r=1}^n$ para todo $i \in \{1, ..., M\}$ y del número de clústeres $K$. Construimos a partir del conjunto $S$ la matriz $X$. El algoritmo $K$-POD sigue los siguientes pasos:

1. Se rellena la matriz $X$ con los datos incompletos utilizando un método que computacionalmente sera barato. Por ejemplo, se pueden rellenar todos los campos de una columna con la media de los elementos no vacíos de esa columna. Después, se ejecuta el algoritmo $K$-means sobre esa matriz y se obtiene para cada elemento el clúster al que pertenece y los centros de los clústeres. Al final, se tienen las matrices $A^{(0)}$ y $B^{(0)}$.

2. Rellenamos los valores vacíos de la matriz $X$ con los valores del clúster al que pertenece:

$$X = P_{\Omega}(X) + P_{\Omega^C}(A^{(m)}B^{(m)}))$$

3. Actualizamos los valores de las matrices $A$ y $B$. Para ello aplicamos el algoritmo $K$-means a los datos completos $X^{(m)}$:

$$(A^{(m+1)}, B^{(m+1)}) = \text{K-Means}(X^{(m)})$$

4. Repetir pasos $2 - 4$ hasta que ningún elemento cambie de clúster asignado.

### 3.1.5. *K*-medoids

Una de las desventajas más importantes del algoritmo $K$-means es su sensibilidad a los *outliers* ya que estos pueden distorsionar la determinación de los centros de los diferentes clústeres. Como solución a este problema surgió una nueva versión del algoritmo llamada $K$-medoids. El algoritmo más representativo de esta nueva versión es el llamado PAM [20] (*Partition Around Medoids*) que es el que se detallará a continuación.

#### Base teórica

La idea básica del algoritmo reside en tomar como centro del clúster un punto real del conjunto de datos en lugar de tomar el valor medio como se hace en el $K$-means. Esto se traduce en que las funciones objetivo son parecidas pero no iguales:

$$E = \sum_{i=1}^{K} \sum_{x_i \in C_i} d\left(x_i, c_i\right) \tag{3.5}$$

donde $c_i$ es el punto tomado como centro del cluster $C_i$, que como ya hemos dicho debe ser un punto del conjunto de datos, $x_i$ son todo los puntos del dataset y $d$ es la función distancia entre dos puntos. La base teórica del algoritmo es similar a la del algoritmo $K$-means: la finalidad es minimizar la suma de las distancias entre los elementos del clúster y maximizar la distancia entre los elementos de los otros.

#### Algoritmo

La ejecución del algoritmo sigue pasos parecidos a los del algoritmo $K$-means.

1. Calcula la distancia entre todos los pares de puntos, basándose en la función de distancia escogida previamente. A partir de esto, calculamos los valores $v_j$, con $1 \leq j \leq n$, de la siguiente forma:
$$v_j = \sum_{i=1}^{n} \frac{d(x_i, x_j)}{\sum_{l=1}^{n} d(x_i, x_l)}$$

2. Selecciona los $K$ objetos que tienen el valor $v_j$ más pequeño y los asigna como centroides que forman clústeres de un único elemento.

3. Asigna a cada objeto del dataset el clúster que tiene el centroide más cercano a dicho objeto.

4. De forma aleatoria, selecciona un punto $x_j$ de cada clúster y, para cada valor $1 \leq i \leq K$, calcula el coste de la función
$$\sum_{x_i \in C_i} d(x_i, x_j)$$

5. Si el coste de esta función con el nuevo punto es menor que el coste calculado con el centroide anterior, se cambia el centroide al nuevo punto. De esta forma, se intenta optimizar las funciones 3.3 y 3.4. Es decir, si se cumple
$$\sum_{x_i \in C_i} d(x_i, x_j) < \sum_{x_i \in C_i} d(x_i, c)$$
entonces el nuevo centro pasa a ser el punto $x_j$.

6. Repetimos los pasos 3, 4 y 5 hasta que se estabilicen los clústeres.

## 3.2. Clústering aglomerativo

Los algoritmos de *clústering aglomerativo* son una de las dos subfamilias de métodos dentro de los algoritmos de clústering jerárquico. La otra subfamilia se conoce como algoritmos divisivos. En este trabajo nos centraremos en la primera de ellas y, a continuación, detallaremos sus principales características [19].

Los métodos aglomerativos tienen por objetivo agrupar datos para formar nuevos clústeres. Sin embargo, los datos no son agrupados en grupos en un único paso. En lugar de eso, tienen lugar una serie de uniones anidadas que desembocan en una agrupación de todos los datos en un único clúster, habiendo partido de $n$ clústeres formados por un único dato cada uno. Tanto los algoritmos aglomerativos como los divisivos se suelen representar de una forma fácil y sencilla como dendogramas o n-árboles, que son unas estructuras que muestran como de difícil es la fusión de dos clústeres, es decir, tratan de determinar cuánto se parecen los clústeres. El nodo raíz representa todo el conjunto de datos y sus hijos representan subconjuntos del conjunto padre. Por ello, las hojas del n-árbol son los puntos del conjunto de datos. En la figura 3.1 se puede ver el funcionamiento de este tipo de métodos. Aunque no

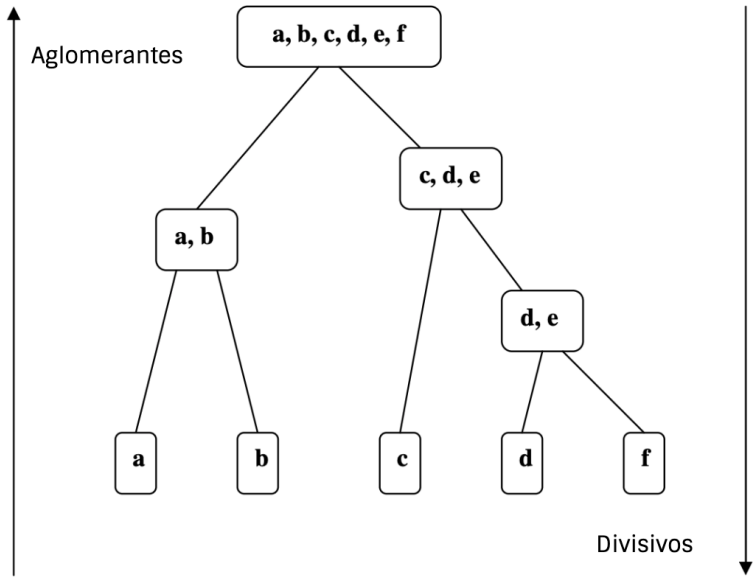



Figura 3.1: Algoritmos aglomerativos y divisivos.

explicaremos con detalle los métodos divisivos, en la misma figura se puede ver que siguen un funcionamiento opuesto al de los algoritmos conglomerantes.

Esta familia de algoritmos fue diseñada para construir un mecanismo más determinista y flexible para la agrupación de datos debido a sus importantes ventajas. Una de ellas, además de su fácil representación y visualización, es que una vez que el dendograma está generado, no es necesario volver a realizar toda la ejecución del algoritmo para obtener una nueva partición de los clústeres. Lo único que hay que hacer es "cortar" el árbol en otro nivel, para así obtener un nuevo número de clústeres y una nueva disposición final de los datos. Este proceso provoca que surja una gran ventaja con respecto a otros algoritmos: no es necesario especificar de antemano el número final de clústeres. Sin embargo, también existen algunas desventajas como que los algoritmos están enfocados para tipos de datos numéricos y para conjunto de datos no muy grandes.

### Base teórica

Sea $X_n$ un conjunto de $n$ datos. Empezamos con $C_1, \ldots, C_n$ clústeres y vamos fusionándolos hasta llegar a tener uno único. La complejidad del algoritmo reside en seleccionar los clústeres que se van a fusionar. En el fondo, todos estos algoritmos fusionan los clústeres que están más cerca unos de otros porque quieren minimizar la función 3.3 y maximizar la función 3.4. Sin embargo, existen diversos métodos para descubrir qué clústeres están más cerca entre sí. A continuación detallaremos los métodos más utilizados:

1. Algoritmos de enlace único. Es uno de los algoritmos jerárquicos más simples. Esta clase de algoritmos son también conocidos por otros nombres como el método del

vecino más cercano o el método mínimo. Emplean la función de distancia del vecino más cercano para medir la distancia entre dos grupos $C_1$ y $C_2$.

$$D_{nn}(C_1, C_2) = \min_{1 \leq i \leq r, 1 \leq j \leq s} d(x_i, x_j) \tag{3.6}$$

donde $x_i \in C_1$, $x_j \in C_2$ y $d$ es la función de distancia entre dos puntos. Una clara desventaja de este método es que es bastante sensible a outliers.

2. Algoritmos de enlace completo. Utiliza la función de distancias del vecino más lejano.

$$D_{fn}(C_1, C_2) = \max_{1 \leq i \leq r, 1 \leq j \leq s} d(x_i, x_j) \tag{3.7}$$

donde $x_i \in C_1$, $x_j \in C_2$ y $d$ es la función de distancia entre dos puntos. Nótese que esta aproximación es equivalente a elegir los dos clústeres cuya fusión crea el menor diámetro. Por este motivo, generalmente generan grupos compactos.

3. Algoritmos de media grupal. Aquí se define la distancia como la media de las distancias de todos los posibles pares de puntos de los dos clústeres.

$$D_{gm}(C_1, C_2) = \frac{\sum_{i,j=1}^{n,m} d(x_i, x_j)}{n \cdot m} \tag{3.8}$$

donde $x_i \in C_1$, $x_j \in C_2$, $d$ es la función de distancia entre dos puntos, $n$ es el número de puntos del clúster $C_1$ y $m$ el número de puntos del clúster $C_2$.

**Algoritmo**

Dado un conjunto de puntos $X$, el primer paso es establecer una función de distancia $d(x_i, x_j)$ entre dos de dichos puntos. Una vez que está definida, se siguen estos pasos:

1. Se crea una matriz de disimilitud:

$$M_{dist}(D) = \begin{pmatrix} 0 & D_{12} & \cdots & D_{1n} \\ D_{21} & 0 & \cdots & D_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ D_{k1} & D_{k2} & \cdots & 0 \end{pmatrix} \tag{3.9}$$

donde $D_{ij} = D(C_i, C_j)$ refleja la distancia entre el cluster $i$ y el clúster $j$ mediante el método de enlace y la función distancia elegida previamente. A su vez todos los puntos son representados en el dendograma en la parte inferior.

2. Utilizando la matriz de disimilitud, se fusionan los clústeres más cercanos.

3. Aplicando la distancia entre clústers elegida previamente, se actualiza la matriz de disimilitud con los clústeres resultantes.

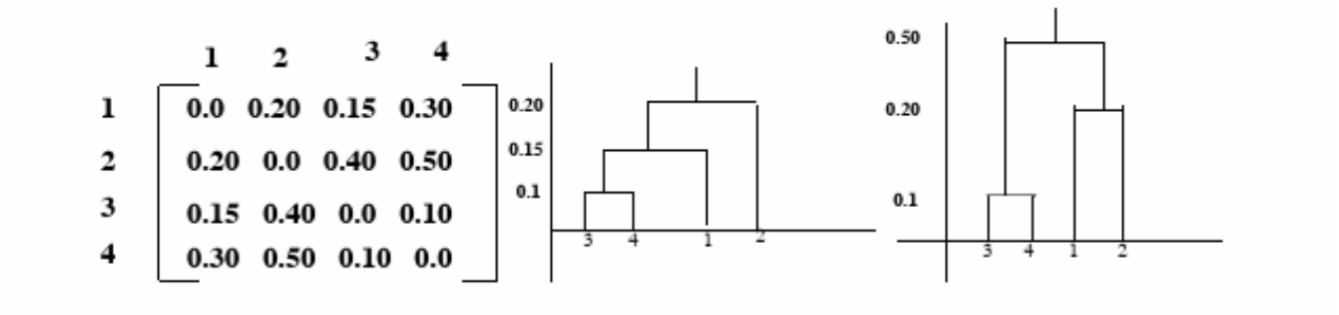



Figura 3.2: Matriz de disimilitud, dendograma de un algoritmo de enlace único y dendograma de un algoritmo de enlace completo

4. Se repiten los pasos 2 y 3 hasta que se tiene un único clúster que engloba todos los puntos.

5. El último paso consiste en "cortar" el dendograma y dividir el conjunto de datos en los grupos finales. Esto se suele hacer mediante un parámetro llamado *distancia umbral* que indica una distancia máxima a partir de la cual los clústeres ya no se fusionan.

En la figura 3.2 se pueden apreciar los diferentes dendogramas resultantes para un dataset de 4 puntos, siguiendo un algoritmo con un método de enlace simple y otro con enlace completo a partir de la misma matriz de disimilitud.

## 3.3. Clústering espectral

Este tipo de clústering [12; 16] surge a partir de la teoría de grafos y está especialmente recomendado para aquellos casos en los que los datos no forman grupos esféricos, sino que siguen otros tipos de patrones.

### Base teórica

Representaremos los puntos de nuestro espacio como un grafo no dirigido valorado, $G = (V, E)$, donde los nodos del grafo son los puntos de nuestro espacio y las aristas que unen cada par de vértices $i$, $j$ tienen un peso $w(i, j)$ que denota la similitud entre el vértice $i$ y el vértice $j$ estos pesos vienen dados por $W$ que es la matriz de afinidad simétrica.

Cuando intentamos agrupar los vértices en conjuntos disjuntos $V_1, V_2, ..., V_K$, lo que buscamos es que la similitud entre los vértices de cada conjunto $V_i$ sea lo mayor posible, y que la medida de similitud entre los vértices de diferentes conjuntos $V_i$, $V_j$ sea lo más pequeña posible.

Para tratar de encontrar la partición óptima, se define el problema del *corte en grafos*, es decir, dado un grafo $G = (V, E)$, dividir el conjunto de vértices en conjuntos disjuntos $A$ y $B$. El grado de similitud entre $A$ y $B$ viene dado por la suma total de los valores de las

aristas que se han tenido que eliminar para llevar a cabo la partición. Estamos interesados en disminuir este valor y, por lo tanto, el problema del corte en grafos consiste en minimizar la función

$$cut(A, B) = \sum_{u \in A, v \in B} w(u, v)$$

En [24] se demostró que este criterio favorece cortar el grafo en pequeños conjuntos aislados pudiendo, por ejemplo, hacer una partición donde uno de los conjuntos tenga un único vértice. Es por eso que se pide de forma natural que el tamaño de los conjuntos sea lo suficientemente grande. Para ello, tendremos en cuenta el peso de las aristas que unen cada uno de los conjuntos con el resto de vértices de $V$, obteniendo la función

$$Ncut(A, B) = \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(A, B)}{assoc(B, V)}$$

donde $assoc(X, V) = \sum_{u \in X, t \in V} w(u, t)$. Se ve fácilmente que el elegir un conjunto pequeño aislado no asegura el tener un valor pequeño de $Ncut$.

Ahora, además de una medida de similitud entre los conjuntos de la partición $A$ y $B$, podemos definir una medida de asociación interna de cada uno de los conjuntos que forman la partición:

$$Nassoc(A, B) = \frac{assoc(A, A)}{assoc(A, V)} + \frac{assoc(B, B)}{assoc(B, V)}$$

Esta medida muestra cuan fuertes son las uniones entre los vértices de cada uno de los conjuntos de la partición, estando interesados en que esta cantidad sea lo más grande posible.

**Lema 3.3.1.** *El valor que minimiza $Ncut$ coincide con el que maximiza $Nassoc$*

La demostración del lema anterior se puede encontrar en [12], a la vista del resultado sabemos que cuando obtengamos la solución óptima para el problema $Ncut$ también habremos encontrado la solución óptima para $Nassoc$.

**Lema 3.3.2.** *Encontrar la solución de $Ncut$ es una problema $NP$-completo*

La demostración de este lema se encuentra también en [12], por lo que resulta interesante trabajar con una aproximación de este problema en el dominio de los reales que se pueda resolver más eficientemente.

**Lema 3.3.3.** *Minimizar el problema Ncut es equivalente a minimizar el sistema de autovalores:*

$$(D - W)y = \lambda Dy$$

*Donde: las coordenadas de $y \in \mathbb{R}^M$, $y^T D\mathbf{1} = 0$, $D$ es la matriz diagonal $M \times M$ (siendo $M$ el número de elementos en el espacio y por tanto el número de nodos del grafo) donde el elemento en la posición $(i, i)$ es $\mathbf{d}_i = \sum_j w(i, j)$, $W$ la matriz de afinidad simétrica $M \times M$ donde el elemento en la posición $(i, j)$ es $w_{i,j}$.*

*De hecho el valor que minimiza este problema es el segundo menor autovalor.*

De nuevo la demostración de este hecho se puede encontrar en [12]

#### Algoritmo

Dados un conjunto de puntos, la matriz de afinidad (aquella que por cada par de vértices indica su similitud) y el número de clústeres que se quieren obtener se llevan a cabo los siguientes pasos:

1. Resolver el sistema $(D - W)x = \lambda Dx$, para obtener el autovector asociado al segundo autovalor más pequeño ($\lambda$).

2. Utilizar el autovector anterior para particionar el grafo en dos partes. Para hacerlo existen muchas posibilidades y, entre las más utilizadas, podemos destacar tomar 0 como punto de corte (es decir, si el autovector tiene en la coordenada $i$ un valor mayor que 0 entonces el punto $v_i$ pertenece al conjunto $A$, en caso contrario pertenece al conjunto $B$), tomar la mediana entre los valores del vector como el punto de corte, o buscar el valor que minimice $Ncut(A, B)$.

3. Iterar el proceso en cada una de las partes de la partición actual hasta que se hayan conseguido los clústeres necesarios.

## 3.4. Algoritmos basados en densidad

Muchos de los algoritmos vistos anteriormente dan por supuesto que los datos generan grupos esféricos y no pueden tratar bien con los clústeres que no siguen este patrón. Sin embargo, es un caso común que la agrupación de los datos tenga formas arbitrarias. Los algoritmos basados en densidad permiten la agrupación de los datos sin la necesidad de seguir una forma concreta y sin especificar de antemano el número final de clústeres. Los algoritmos basados en densidad agrupan áreas densas y conectadas en el conjunto de datos separadas entre sí por zonas de datos más diluidas. Además, se supone que la densidad dentro de las áreas de ruido es menor que la densidad en cualquiera de los grupos. De esta forma, las zonas menos densas de los datos, son clasificadas como outliers y no son asignadas a ningún clúster. Entre todos los algoritmos de este tipo detallaremos los algoritmos DBSCAN y OPTICS.

### 3.4.1. DBSCAN

Es el algoritmo de clústering de densidad más usado. Su nombre es un acrónimo de las palabras en inglés *Density Based Spatial Clustering of Applications with Noise*. La idea del algoritmo consiste en agrupar los puntos en el mismo clúster si forman un grupo denso. Este concepto será explicado a continuación.

#### Base teórica

Este algoritmo utiliza un parámetro de *umbral de densidad global* llamado $\epsilon$. Es decir, para todo el conjunto de datos existe una misma noción para determinar si un subconjunto

de datos es denso [3]. Ello puede provocar que en algunas ocasiones, el algoritmo no se adapte adecuadamente al conjunto de datos y no genere los clústeres correctamente. No obstante, el algoritmo necesita recibir otro parámetro que indica un número mínimo de puntos $N_{min}$. Pasamos a continuación a presentar algunos conceptos que se usarán más adelante.

**Definición 3.4.1.** *Sea $S$ un conjunto de datos, $d$ la función de similitud, $N_{min}$ un valor menor que $|S|$ y $\epsilon > 0$. Dado $x \in S$, definimos su* $\epsilon$-vecindad *como*

$$N_\epsilon(x) = \{y \in S | d(x, y) \leq \epsilon\} \tag{3.10}$$

*Dados $x, y \in S$, decimos que $y$ es* directamente alcanzable por densidad *desde $x$ si $y \in N_\epsilon(x)$ y se cumple que el número de puntos pertenecientes a $N_\epsilon(x)$ es mayor que $N_{min}$.*

*La noción anterior se puede adaptar para que sea simétrica. Dados $x, y \in S$, decimos que están* conectados por densidad *con respecto a $\epsilon$ y a $N_{min}$ si existe un punto $z$ tal que tanto $x$ como $y$ son directamente alcanzables por densidad desde $z$ con respecto a $\epsilon$ y a $N_{min}$.*

Una vez presentadas las nociones anteriores, podemos dar la definición técnica de clúster basado en densidad.

**Definición 3.4.2.** *Sea $S$ un conjunto de datos, $N_{min} < |S|$ y $\epsilon > 0$. Un* clúster basado en densidad *$C$ respecto a $\epsilon$ y $N_{min}$ tiene que cumplir las siguientes condiciones:*

1. *$\emptyset \neq C \subset S$.*
2. *Para todo $x, y \in S$ si $x \in C$ e $y$ es alcanzable por densidad desde $x$ con respecto a $\epsilon$ y $N_{min}$, entonces $y \in C$. Esto genera maximalidad en los conjuntos*
3. *Para todo $x, y \in C$, $x$ e $y$ están densamente conectados con respecto a $\epsilon$ y $N_{min}$.*

Siguiendo las definiciones previas se pueden establecer tres tipos de puntos

- *Punto central.* Son aquellos cuya $\epsilon$-vecindad tiene más puntos que el mínimo de puntos.
- *Puntos frontera.* Son aquellos que pertenecen a un clúster pero no son puntos centrales.
- *Outliers.* Son aquellos puntos que no pertenecen a ningún clúster.

### Algoritmo

Una vez definidos los conceptos anteriores, podemos presentar las cuatro fases que constituyen el algoritmo.

1. En primer lugar, todos los puntos están marcados como "no visitado".
2. Se selecciona un punto $x$ del conjunto de puntos del dataset de forma arbitraria. Se marca como "visitado" y se trata de encontrar todos los puntos del dataset que son densamente alcanzables desde $x$ con respecto a $\epsilon$ y $N_{min}$.

3. Si $x$ es un punto central entonces se forma un clúster y se marca como tal. Si no lo es, puede ser un punto frontera o outlier. En ambos casos, se marca como outlier ya que ningún punto es densamente alcanzable desde $x$. Esta clasificación no es definitiva ya que cuando posteriormente se analice otro punto $y$ que sea central, se podrá determinar si $x$ pertenece a la $\epsilon$-vecindad de este nuevo punto $y$, deduciéndose en caso afirmativo que se trata de un punto frontera.

4. Una vez que se ha analizado un punto y se ha clasificado, se selecciona aleatoriamente otro punto marcado como "no visitado" y se repiten los pasos 2 y 3. El algoritmo termina cuando todos los puntos han sido evaluados.

### 3.4.2. OPTICS

La idea básica del algoritmo OPTICS reside en tratar de agrupar los puntos según formen grupos densos pero, a diferencia del algoritmo DBSCAN, se trata la densidad de forma local en diferentes subconjuntos del dataset. De esta forma, el algoritmo se puede adaptar mejor a la estructura del conjunto de datos y puede obtener mejores resultados [1; 2]. Sin embargo, OPTICS también tiene que recibir como parámetros $\epsilon$ y $N_{min}$, pero no tiene la misma función que en el algoritmo anterior. De hecho, en este caso el valor $\epsilon$ se suele tomar mucho más grande. Aunque OPTICS se apoye en definiciones vistas en el algoritmo DBSCAN, existen diferencias notables que hacen interesante el estudio de este nuevo algoritmo.

#### Base teórica

La idea básica de este algoritmo reside en crear una lista ordenada de los datos siguiendo un método que se explicará a continuación. En esta lista, los datos más cercanos entre sí se encontrarán en posiciones más cercanas. A partir de dicha lista, se elabora posteriormente la clasificación de los clústeres. Presentamos dos conceptos que permitirán explicar el funcionamiento del algoritmo.

**Definición 3.4.3.** *Sea $S$ un conjunto de datos, $N_{min} < |S|$ y $\epsilon > 0$. La* distancia núcleo $\epsilon'$ *de un objeto $x \in S$ es el valor más pequeño que hace que $x$ sea un punto central (siguiendo la definición vista en la definición 3.4.2). Por el contrario, si $x$ no es un punto central con respecto a $\epsilon$ y $N_{min}$, entonces la distancia-núcleo de $x$ no está definida.*

*Sean $x, y \in S$. La* distancia alcanzable *entre $x$ e $y$ es el mayor valor entre la distancia núcleo de $x$ y la distancia euclídea entre $x$ y $y$ o no está definida si $x$ no es un punto central con respecto $\epsilon$ y $N_{min}$.*

En la figura 3.3 se puede ver reflejado el uso de las dos definiciones anteriores. Considerando $N_{min} = 4$, $\epsilon = 6$, obtenemos que $\epsilon' = 3$.

#### Algoritmo

Como se ha explicado antes, el objetivo del algoritmo es elaborar una lista ordenada de puntos y a partir de dicha lista establecer el agrupamiento. Para ello se siguen los siguientes

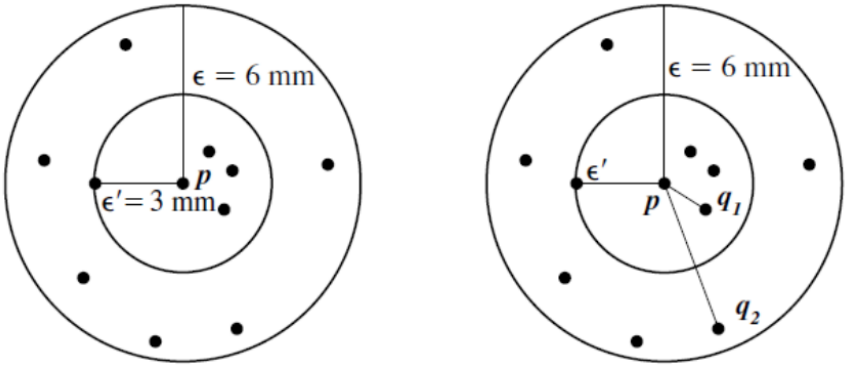



Figura 3.3: Distancia alcanzable entre $p$ y $q_1$ es $\epsilon$'. Distancia alcanzable entre $p$ y $q_2$ es la distancia entre $p$ y $q_2$

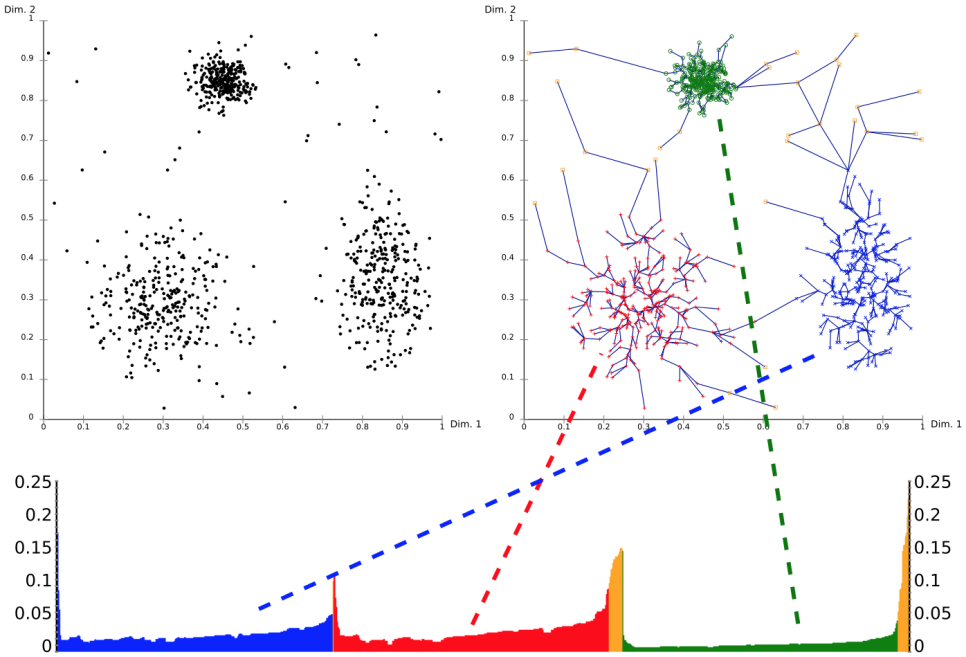



Figura 3.4: Diagrama de alcanzabilidad

pasos:

1. Se marcan todos los puntos como "no visitado" antes de la primera iteración.

2. Se selecciona un punto $x$ "no visitado" de forma aleatoria, se marca como "visitado" y se guarda en qué orden fue procesado. Se estudia si este punto es un punto central respecto a $N_{min}$ y a $\epsilon$. Si no lo es, simplemente se selecciona otro punto "no visitado". En cambio, si $x$ es un punto central se determina la distancia al núcleo $\epsilon'$ y se intenta expandir el clúster.

3. Expansión de clúster. Todos los puntos que son directamente alcanzables por densidad son insertados en la lista ordenada, siguiendo la distancia alcanzable con respecto a punto central $x$. El siguiente paso es seleccionar el punto de esta lista ordenada que tenga menor distancia alcanzable, es decir, el primero de la lista que no haya sido visitado.

4. Se comprueba si este nuevo punto es un punto central o no. De no serlo, se selecciona el siguiente de la lista. De serlo, se aplica el paso 3 y se guarda el orden en el que fue procesado.

5. Si todos los puntos de la lista ordenada han sido visitados, se repite el proceso empezando en el paso 2. Este proceso termina cuando todos los puntos han sido visitados.

6. Proceso de agrupación. Una vez que todos los puntos se han visitado, hay que separar los clústeres. Para ello se construye un diagrama de alcanzabilidad a partir de la lista ordenada y del orden en el que fueron procesados (véase la figura 3.4). El eje $x$ del diagrama indica el orden en el que fue procesado cada punto mientras que el eje $y$ guarda su distancia alcanzable al punto central más cercano. Nótese que cuanto más denso es un clúster, menores serán las distancias de alcanzables y más bajo es el valle en el diagrama. Los clústeres menos densos presentan distancias de alcanzables mayores y valles más altos en el diagrama. Por último, los picos representan las distancias que es necesario recorrer para viajar de un clúster a otro, o de un clúster a ruido. A partir de este diagrama, y con la ayuda de un parámetro, se "corta" el diagrama cuando aparece un punto de ruido y se establecen los clústeres.

# Capítulo 4

# Base teórica de los métodos de evaluación

En la fase de evaluación tenemos dos objetivos: determinar si los datos con los que estamos trabajando son adecuados para aplicar métodos de clústering y determinar cómo de buenos son los algoritmos de clústering aplicados. En este capítulo presentaremos en primer lugar el método que utilizaremos para evaluar nuestro dataset y en segundo lugar los métodos para evaluar los algoritmos utilizados. Todos estos métodos se pueden encontrar en [9].

## 4.1. Evaluación del dataset

Al evaluar el dataset buscamos determinar que la estructura de nuestro conjunto de datos no es aleatoria, pues en el caso de que lo sea los clústeres obtenidos tras la aplicación de los métodos no van a ser relevantes. Utilizaremos el *estadístico de Hopkins* para determinar la aleatoridad con la que los datos de nuestro dataset están distribuidos en el espacio. Dada una variable aleatoria $x$, buscamos determinar cuán alejada está $x$ de estar distribuida de manera uniforme en el espacio. A partir de un conjunto $S = \{x_1, ..., x_n\}$, para calcular este estadístico tendremos que:

1. Generar $m$ puntos $y_1, y_2, ..., y_m$ distribuidos de forma uniforme en el espacio. Es importante destacar que estos valores no tienen por qué pertenecer a $S$. Para cada punto $y_i$, calculamos su vecino $x_j \in S$ más cercano y calculamos dicha distancia, a la que llamaremos $u_i = dist(y_i, x_j)$.

2. Generar $\bar{x}_1, ..., \bar{x}_m$ una muestra aleatoria (sin reemplazamiento) de los puntos de $S$. Calculamos para cada punto $\bar{x}_i$ el vecino más cercano $x_j \in S$ y calculamos su distancia, a la que llamaremos $w_i = dist(\bar{x}_i, x_j)$.

3. El estadístico de Hopkins se calcula como:

$$H = \frac{\sum_{i=1}^{n} w_i}{\sum_{i=1}^{n} w_i + \sum_{i=1}^{n} u_i}$$

El valor de este estadístico estará entre $[0, 1]$. Nótese que los datasets que tengan sus datos distribuidos de forma uniforme tendrán un valor $H = 0{,}5$ porque los valores $u_i$ y $w_i$ serán parecidos. Por el contrario, si los datos están formando clústeres, los valores de $u_i$ y $w_i$ serán muy distintos. En concreto, $w_i$ será típicamente mucho más pequeño que $u_i$ y por ello en el caso de tener un dataset idóneo para clústering tendremos $H$ cercano a 0.

En la literatura es también muy común encontrarse el estadístico de Hopkins enunciado como:

$$H = \frac{\sum_{i=1}^{n} u_i}{\sum_{i=1}^{n} w_i + \sum_{i=1}^{n} u_i}$$

En este caso, los datos idóneos para clústering tendrán un valor de $H$ cercano a 1.

Nosotros utilizaremos el primer enfoque ya que es el seguido por la implementación [1] del estadístico que hemos utilizado.

## 4.2. Evaluación de los métodos de clústering

Después de aplicar un algoritmo de clústering sobre los datos, surge de manera natural la siguiente pregunta ¿cómo de buenos son los resultados obtenidos? Además, una vez que se han aplicado varios métodos nos podemos preguntar ¿cómo podemos comparar estos métodos entre sí? Para dar respuesta a estas dos preguntas tenemos distintas posibilidades. Por un lado, están los denominados métodos *extrínsecos* que son aquellos que utilizan el clústering ideal para determinar cómo de bueno es el método empleado. El clústering ideal se obtiene normalmente de expertos: en nuestro caso, el clústering ideal coincide con el diagnóstico que aparece en los datos. En contraposición, los métodos *intrínsecos* son aquellos que no disponen de esa información.

### 4.2.1. Métodos intrínsecos

Estos métodos evalúan como de bueno es un método de clústering examinando como de separados estás los clústeres entre sí y cómo de compactos son. En nuestro caso, utilizaremos

[1] https://pyclustertend.readthedocs.io/en/master/

este tipo de métodos además de para comparar los distintos algoritmos entre sí, para determinar el número de clústeres en aquellos métodos que lo necesiten como parámetro como es el caso de $K$-means. El método que emplearemos es uno de los más conocidos y utilizados: el *Coeficiente de Silhouette*. Dado un dataset $S$ con $n$ elementos, supongamos que este ha sido dividido en $K$ clústeres $C_1, C_2, ..., C_K$. Para cada elemento $x \in S$, calculamos $a(x)$ como la distancia media entre $x$ y todo el resto de elementos del clúster al que $x$ pertenece. Intuitivamente, $a(x)$ representa lo compacto que es el clúster al que $x$ pertenece: cuando más pequeño sea el valor de $a(x)$ más compacto será. De forma similar, $b(x)$ es la mínima distancia media del elemento $x$ a todos los clústeres a los que $x$ no pertenece. Por tanto, el valor de $b(x)$ representa lo separado que está el elemento de otros clústeres, Formalmente, si suponemos que $x \in C_i$, entonces:

$$a(x) = \frac{\sum_{x' \in C_i, x \neq x'} dist(x, x')}{|C_i| - 1}$$

$$b(x) = \min_{1 \leq j \leq K, j \neq i} \left\{ \frac{\sum_{x' \in C_j} dist(x, x')}{|C_j|} \right\}$$

Finalmente, el coeficiente de Silhouette para un elemento se calcula como:

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

El valor del coeficiente de Silhouette está entre $-1$ y 1. A tenor del significado intuitivo de los valores de $a$ y $b$, cuando el valor del coeficiente de Silhoueete se acerca a 1 significa que el clúster es compacto y que está separado del resto (la situación más deseable), mientras que si el coeficiente de Silhouette es negativo ($b(x) < a(x)$), el elemento $x$ está más cerca de los elementos de otros clústeres que de los de su propio clúster.

El coeficiente de Silhouette para el dataset se define como la media de los coeficientes para todos los elementos, es decir como:

$$S = \frac{1}{n} \cdot \sum_{j=1}^{K} \sum_{x \in C_j} s(x)$$

### 4.2.2. Métodos extrínsecos

En este tipo de métodos tendremos que comparar los resultados obtenidos tras aplicar el método de clústering con el clústering ideal. Por lo tanto, si el clústering obtenido por el algoritmo es $C$ y el clústering ideal es $C_i$ tendremos que determinar una medida $Q$ que de una puntuación adecuada a $Q(C, C_i)$. Intuitivamente, valores altos de $Q$ indican que $C$ es un clústering mejor. Esta medida debe cumplir los siguientes criterios:

- *Homogeneidad de los clústeres.* Esto significa que cuanto más puro sea un clúster mejor. Por ejemplo, supongamos que las clases del clústering ideal son $L_1, ..., L_m$. Consideremos un clústering $\mathcal{C}_1$ donde existe un clúster $C \in \mathcal{C}_1$ que contiene elementos de dos categorías $L_i, L_j$, con $i \neq j$, y consideremos también un clústering $\mathcal{C}_2$ que es idéntico al primero excepto porque el clúster $C$ se divide en dos de forma que cada uno de los clústers contiene únicamente elementos de la clase $L_i$ o $L_j$. Entonces, $Q$ debe cumplir $Q(\mathcal{C}_2, C_i) > Q(\mathcal{C}_1, C_i)$.

- *Completitud de los clústeres.* Esto significa que si dos elementos están en la misma clase en el clústering ideal, es deseable que también estén en el mismo clúster en nuestro clústering. Consideremos un clústering $\mathcal{C}_1$ que contiene clústeres $C_1$ y $C_2$ en los cuales todos los miembros de ambos pertenecen a la misma categoría según el clústering ideal. Sea $\mathcal{C}_2$ otro clústering idéntico a $\mathcal{C}_1$ excepto porque los clústeres $C_1$ y $C_2$ están juntos formando un único clúster. Entonces $Q$ debe cumplir $Q(\mathcal{C}_2, C_i) > Q(\mathcal{C}_1, C_i)$.

- *Clúster heterogéneo ("Rag Bag").* Se debe penalizar más el incluir un elemento heterogéneo en un clúster homogéneo que introducirlo en un clúster heterogéneo. Por ejemplo, consideremos un clústering $\mathcal{C}_1$ que tiene un clúster $C$ donde todos los elementos excepto un elemento $x$ tienen la misma clase. Consideremos también otro clústering $\mathcal{C}_2$ idéntico a $\mathcal{C}_1$ excepto porque el elemento $x$ es asignado a un clúster $C' \neq C$ que contiene elementos de distintas clases. Entonces $Q$ debe cumplir $Q(\mathcal{C}_2, C_i) > Q(\mathcal{C}_1, C_i)$.

- *Preservación de los clústeres pequeños.* Si una categoría con pocos elementos se divide en piezas pequeñas es muy probable que esa categoría no pueda ser descubierta a partir del clústering. Por lo tanto, se debe penalizar más dividir en varios clústeres una categoría con pocos elementos que con muchos elementos. Consideremos por ejemplo un dataset $S$ con $n+2$ elementos, en el clústering ideal $n$ de ellos (sin pérdida de generalidad, supongamos que son los $n$ primeros) pertenecen a una categoría y $x_{n+1}, x_{n+2}$ pertenecen a otra categoría. Supongamos ahora que tenemos un clústering $\mathcal{C}_1$ que tiene tres clústeres $C_1 = \{x_1, ..., x_n\}$, $C_2 = \{x_{n+1}\}$, $C_3 = \{x_{n+2}\}$. Y sea $\mathcal{C}_2$ otro clústering que tiene tres clústeres $C_1 = \{x_1, ..., x_{n-1}\}$, $C_2 = \{x_n\}$, $C_3 = \{x_{n+1}, x_{n+2}\}$. Entonces $Q$ debe cumplir $Q(\mathcal{C}_2, C_i) > Q(\mathcal{C}_1, C_i)$.

El valor-F *BCubed* da lugar a una métrica que satisface todos los criterios previamente descritos. Utilizaremos la implementación en python[2] de este valor. Pasamos a definir formalmente el proceso para calcularlo.

**Definición 4.2.1.** *Sea $S = \{x_1, ..., x_n\}$ un dataset, $\mathcal{C}$ la agrupación en clústeres de $S$, $L(x_i)$ la categoría asignada por el clústering ideal a cada elemento de $x_i \in S$ y $C(x_i) \in \mathcal{C}$ el clúster al que pertenecen cada uno de los elementos. Definimos la* corrección *como una relación entre dos elementos distintos del dataset $x_i, x_j \in S$, $i \neq j$, de la siguiente forma:*

$$Corrección(x_i, x_j) = \begin{cases} 1, & si\ L(x_i) = L(x_j)\ y\ C(x_i) = C(x_j) \\ 0, & en\ otro\ caso \end{cases}$$

[2] https://github.com/m-wiesner/BCUBED

*La* precisión *para un elemento indica cuántos elementos que están en el mismo clúster que él pertenecen a la misma categoría, la* precisión BCubed *evalúa la precisión de todos los elementos del dataset:*

$$\textit{Precisión BCubed}(\mathcal{C}) = \frac{1}{n} \cdot \sum_{i=1}^{n} \frac{\sum\limits_{\substack{1 \leq j \leq n, i \neq j, \\ C(x_i) = C(x_j)}} \textit{Corrección}(x_i, x_j)}{|\{x_j | i \neq j, C(x_i) = C(x_j)\}|}$$

*La* exhaustividad *para un elemento indica cuántos elementos de la misma categoría han sido asignados al mismo clúster, la* exhaustividad BCubed *evalúa la exhaustividad de todos los elementos del dataset:*

$$\textit{Exhaustividad BCubed}(\mathcal{C}) = \frac{1}{n} \cdot \sum_{i=1}^{n} \frac{\sum\limits_{\substack{1 \leq j \leq n, i \neq j, \\ L(x_i) = L(x_j)}} \textit{Corrección}(x_i, x_j)}{|\{x_j | i \neq j, L(x_i) = L(x_j)\}|}$$

*El valor-F BCubed es un valor ponderado de la precisión BCubed y de la exhaustividad BCubed que se define como:*

$$\textit{Valor-F BCubed}(\mathcal{C}) = \frac{(1 + \beta^2) \cdot \textit{Precisión BCubed}(\mathcal{C}) \cdot \textit{Exhaustividad BCubed}(\mathcal{C})}{(\beta^2 \cdot \ \textit{Precisión BCubed}(\mathcal{C})) + \textit{Exhaustividad BCubed}(\mathcal{C})}$$

# Capítulo 5

# Resultados

En los capítulos anteriores hemos mostrado que podemos considerar 11 datasets distintos y 9 métodos de clustering. Por tanto, en total tenemos 99 combinaciones y debemos decidir cuál es la mejor. Para llevar a cabo esta elección de manera eficiente utilizaremos las técnicas de evaluación de los métodos de clustering (presentadas en las secciones 4.2.1 y 4.2.2) y, basándonos en ellas, decidiremos qué dataset se comporta mejor para cada método. Después compararemos la distribución en clústeres para las 9 combinaciones seleccionadas y estaremos en posición de seleccionar la mejor.

En este capítulo presentaremos, en primer lugar, algunos experimentos previos, después analizaremos la calidad de los datasets utilizando el estadístico de Hopkins, a continuación analizaremos los resultados de cada algoritmo por separado, mostrando para cada uno de ellos una tabla que contenga el valor del coeficiente de Silhouette, el valor-F , la precisión y la exhaustividad Bcubed. Finalmente, mostraremos una tabla donde compararemos las mejores combinaciones y presentaremos nuestras conclusiones.

## 5.1. Pruebas previas

Antes de llevar a cabo las pruebas definitivas, presentamos una serie de experimentos (fallidos) que nos permitieron construir los modelos finales.

### 5.1.1. Fijando el número de clústeres a 2

Durante nuestros experimentos elegíamos el número de clústeres óptimos para aquellos métodos que necesitan hacer esta elección (por ejemplo KMeans, KPrototypes, KMedoids...) para cada algoritmo y Dataset utilizando el coeficiente de Silhouette. Haciendo esto, en la mayoría de experimentos obteníamos que el número óptimo de clústeres era 2. Nos pareció que este resultado era razonable dado que estos clústeres dividían bastante bien a los pacientes etiquetados en pacientes con EC y pacientes sin EC. Sin embargo, en un número no despreciable de experimentos obteníamos un número óptimo de clústeres muy elevado (para algunos casos 20) que no éramos capaces de interpretar puesto que eran clústeres muy pe-

| | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | C10 | C11 | C12 | C13 | C14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Aún en estudio | - | 1 | 5 | 1 | 1 | - | 3 | - | 1 | 4 | 1 | - | - | 4 |
| EC | 1 | 65 | 22 | 1 | - | - | 1 | 44 | 14 | 12 | 1 | - | 42 | 59 |
| EC Potencial | 1 | 1 | 1 | 11 | 8 | 1 | - | - | 2 | - | 1 | 6 | - | 5 |
| EC Refractaria | - | 1 | 1 | - | - | 1 | - | | - | - | - | - | - | 2 |
| EC Dudosa | - | 1 | 1 | 1 | - | 2 | 2 | - | 3 | - | 1 | - | - | 2 |
| Paciente perdido | - | - | - | - | 1 | - | - | - | - | - | - | - | - | - |
| SGNC no estricta | 3 | - | - | - | 5 | - | - | - | 4 | - | 1 | 6 | - | 1 |
| SGNC estricta | - | - | 1 | - | 1 | - | 1 | - | 1 | - | - | - | - | - |
| Sin diagnóstico | 13 | 2 | 3 | - | 4 | 11 | 2 | - | 10 | 37 | 2 | 1 | 2 | 2 |
| no EC ni SGNC | 4 | - | 6 | 1 | 28 | 57 | 4 | - | 18 | 12 | 9 | 12 | 2 | 6 |

Cuadro 5.1: Clustering óptimo $K$-Means con dataset selección de algoritmos CFS

queños que mezclaban individuos y contenían pacientes con diagnósticos muy heterogéneos, como podemos ver en la tabla 5.1. En dicha tabla podemos ver el resultado obtenido al ejecutar el algoritmo KMeans utilizando el dataset con la selección de atributos CFS. Observamos que se ha creado un número muy elevado de clústeres, en total 14, siendo muchos de ellos tan heterogéneos que resulta imposible sacar alguna conclusión de la agrupación.

Teniendo en cuenta estas consideraciones, decidimos que lo mejor era elegir como número fijo de clústeres 2. Al tomar esta decisión fuimos conscientes de que estábamos "dejando de lado" uno de nuestros objetivos principales que era el de encontrar patrones o grupos entre individuos con el mismo diagnóstico. Sin embargo, como ya hemos indicado, no encontramos ningún algoritmo que creara clústeres homogéneos. Al hacer esta elección, nuestra herramienta es capaz únicamente de distinguir entre pacientes celíacos y no celíacos, como si de un clasificador se tratase, obteniendo unos considerablemente buenos resultados. Por este motivo el análisis de los resultados lo haremos como si trabajásemos con un método clasificador y elegiremos como el mejor experimento aquel que tenga un menor número de fallos, es decir, aquel que clasifique bien al mayor número posible de pacientes etiquetados.

### 5.1.2. Mejorando la imputación

Después de una reunión con la médica, donde le enseñamos los resultados que habíamos obtenido hasta el momento y que tuvo lugar cerca de la fecha de entrega de este trabajo, esta nos señaló que la imputación de valores no funcionaba bien. Específicamente, fallaba en el caso de la prueba *LIEs*, que tiene alta especificidad y sensibilidad, e indicó que no funcionaba como esperaba en las pruebas *ATG_2* y *A-PDG*, aunque estas pruebas no son tan específicas ni sensibles como la anterior. Lo que ocurría para la prueba *LIEs* es que los valores imputados no eran correctos en muchos casos para pacientes con un diagnóstico. Por ejemplo, nuestra imputación estaba asignando en muchos casos valores a un paciente con EC que corresponden a un diagnóstico no EC. Por este motivo la imputación pierde el efecto que buscábamos, aunque como comentamos en la discusión de la sección 2.3.2 seguimos considerando que es la mejor opción disponible ante la baja calidad de los datos de los que

| Dataset | Valor |
|---|---|
| Numérico | 0.290 |
| Numérico acortado | 0.270 |
| Numérico sin país, sexo ni edad | 0.259 |
| Numérico sin país, sexo ni edad acortado | 0.231 |
| Numérico sin síntomas ni signos | 0.314 |
| Numérico sin síntomas ni signos acortado | 0.291 |
| Numérico sin país, sexo, edad, síntomas ni signos | 0.269 |
| Numérico sin país, sexo, edad, síntomas ni signos acortado | 0.232 |
| Numérico elementos diagnósticos principales | 0.217 |
| Numérico selección atributos FCBF | 0.007 |
| Numérico selección atributos CFS | 0.117 |

Cuadro 5.2: Estadístico de Hopkins

disponíamos.

A la vista de esto tratamos de obtener mejores resultados en la imputación. Para ello transformamos la imputación básica en la cual utilizábamos todas las columnas del dataset (sin escalar) para obtener la distancia entre los pacientes por una imputación un poco más compleja en la que sólo teníamos en cuenta las columnas consideradas relevantes en el diagnóstico real de la enfermedad[1] y escalamos el valor de dichas columnas para que todas tuvieran el mismo efecto en la distancia final.

Los experimentos con esta imputación mejoraron los resultados obtenidos con la imputación inicial en todos los algoritmos. Los algoritmos $K$-Means y $K$-Medoids son los que más notaron esta mejora, con un aumento del 1 % y del 5,3 %, respectivamente, en su porcentaje de aciertos. Por lo tanto, los resultados que aquí mostraremos corresponden a esta imputación.

## 5.2. Evaluación de los datasets

Como ya hemos explicado anteriormente, el estadístico de Hopkins es la métrica que utilizaremos para determinar cómo de buenos son nuestros datasets. Esta evaluación nos servirá para determinar los datasets que es esperable que mejor se comporten. Es necesario tener en cuenta que este estadístico está implementado únicamente para datasets numéricos, pero nosotros trabajamos también con datasets categóricos, mixtos e incompletos. En estos casos asumiremos que algo similar ocurre para los datasets categóricos, mixtos e incompletos.

Para los datasets numéricos, la tabla 5.2 presenta los resultados obtenidos. Hay que tener en cuenta que las cantidades numéricas pueden variar un poco ya que el algoritmo hace elecciones aleatorias pero en todos los casos el orden de los datasets es siempre el mismo.

[1]Específicamente, *HLA: grupos de riesgo,DCG EMA, Biopsia DCG, Valoracion LIEs DCG, Valoracion LIEs DSG,Biopsia DSG, DCG_ ATG2_ VALUE, DCG A-PDG_ VALUE, 1º grado, LIEs DCG %GD, LIEs DCG %iNK, DSG ATG2 VALUE, DSG A-PDG VALUE, LIEs DSG %GD, LIEs DSG %iNK.*

A la vista de la tabla 5.2, y lo comentado en la sección 4.1, cabe esperar que los algoritmos de clustering se comporten peor con aquellos datasets que tienen un valor del estadístico más cercano a 0.5. Es decir, cabe esperar que los mejores resultados se obtengan con los datasets *con atributos CFS, atributos FCBF, elementos diagnósticos principales y sin país, sexo, edad, síntomas ni signos acortado*.

Hay que tener en cuenta que cuando decimos "se comportan mejor" nos referimos a tener en cuenta la agrupación espacial. Específicamente, nos referimos a los datasets en los que los datos están distribuidos de una forma menos uniforme por el espacio y, por lo tanto, tienen coeficientes de Silhouette mejores. Nótese que esto no significa que el valor-F, y por tanto la clasificación de los pacientes, sea mejor.

## 5.3. Evaluación de los métodos

En esta sección llevaremos a cabo una evaluación del comportamiento de los métodos de clustering sobre cada uno de los datasets. Para cada uno de los métodos mostraremos una tabla donde aparezca el valor máximo del coeficiente de Silhouette, el valor-F, la precisión y la exhaustividad BCubed. En función de estos valores decidiremos con qué dataset se comporta mejor cada método. Podemos adelantar que, puesto que nuestro análisis se basará únicamente en elegir el algoritmo con mayor porcentaje de aciertos entre los individuos etiquetados, el coeficiente de Silhouette no será de utilidad porque seleccionaremos en todos los casos el experimento con mayor valor-F, dado que nos asegura un mayor porcentaje de acierto.

Una vez se haya seleccionado el dataset con el que mejor se comporta cada uno de los métodos, estudiaremos más de cerca los clústeres y qué diagnósticos tienen los individuos de cada uno de ellos. Además, a la hora de analizar los resultados (es decir, de determinar el porcentaje de error de cada experimento) consideraremos que los pacientes con *EC Refractaria* pertenecen a la clase *EC* y los pacientes *SGNC no estricta y SGNC estricta* pertenecen a la clase *no EC*. Los pacientes que tienen como diagnóstico *Aún en estudio, EC Potencial, EC Dudosa, Paciente perdido* y *Sin diagnóstico* no tienen un diagnóstico claro y consideraremos que están bien clasificados estén en el clúster que estén.

### 5.3.1. K-Means

Tras la ejecución del algoritmo sobre cada uno de los datasets, los valores de las métricas de evaluación se pueden ver en la tabla 5.3. A la vista de los valores de las métricas consideramos que el dataset con el que mejor se comporta nuestro algoritmo es *Sin síntomas ni signos*. Podemos ver el desglose detallado de los clústeres en la tabla 5.12. A la vista de los resultados observamos que hay 17 pacientes "mal clasificados" en el clúster con elementos que pertenecen mayoritariamente a la clase *EC* y hay 27 pacientes "mal clasificados" en el clúster con elementos que pertenecen mayoritariamente a la clase *no EC*. Por lo tanto, este algoritmo tiene un porcentaje de fallo del 7,2 %.

| Dataset | Silhouette | valor-F | Precisión | Exhaustividad |
|---|---|---|---|---|
| Original | 0.108 | 0.61 | 0.492 | 0.802 |
| Acortado | 0.082 | 0.61 | 0.491 | 0.804 |
| Sin país, sexo ni edad | 0.14 | 0.609 | 0.491 | 0.8 |
| Sin país, sexo ni edad acortado | 0.116 | 0.602 | 0.485 | 0.793 |
| Sin síntomas ni signos | 0.122 | 0.612 | 0.496 | 0.801 |
| Sin síntomas ni signos acortado | 0.097 | 0.609 | 0.491 | 0.801 |
| Sin país, sexo, edad, síntomas ni signos | 0.169 | 0.605 | 0.49 | 0.793 |
| Sin país, sexo, edad, síntomas ni signos acortado | 0.155 | 0.609 | 0.491 | 0.802 |
| Elementos diagnósticos principales | 0.132 | 0.61 | 0.494 | 0.799 |
| Selección atributos CFS | 0.275 | 0.603 | 0.488 | 0.789 |
| Selección atributos FCBF | 0.496 | 0.557 | 0.439 | 0.761 |

Cuadro 5.3: Resultados de los diferentes datasets con el algoritmos $K$-Means

| Dataset | Silhouette | valor-F | Precisión | Exhaustividad |
|---|---|---|---|---|
| Original | 0.106 | 0.606 | 0.489 | 0.798 |
| Acortado | 0.08 | 0.577 | 0.463 | 0.766 |
| Sin país, sexo ni edad | 0.136 | 0.611 | 0.493 | 0.804 |
| Sin país, sexo ni edad acortado | 0.115 | 0.584 | 0.468 | 0.775 |
| Sin síntomas ni signos | 0.122 | 0.634 | 0.514 | 0.829 |
| Sin síntomas ni signos acortado | 0.093 | 0.598 | 0.482 | 0.786 |
| Sin país, sexo, edad, síntomas ni signos | 0.167 | 0.625 | 0.505 | 0.819 |
| Sin país, sexo, edad, síntomas ni signos acortado | 0.148 | 0.597 | 0.482 | 0.784 |
| Elementos diagnósticos principales | 0.099 | 0.423 | 0.318 | 0.633 |
| Selección atributos CFS | 0.237 | 0.617 | 0.498 | 0.809 |
| Selección atributos FCBF | 0.434 | 0.541 | 0.433 | 0.722 |

Cuadro 5.4: Resultados de los diferentes datasets con el algoritmos $K$-Prototypes

### 5.3.2. K-Prototypes

Tras la ejecución del algoritmo sobre cada uno de los datasets, los valores de las métricas de evaluación se pueden ver en la tabla 5.4. A la vista de los valores de las métricas consideramos que el dataset con el que mejor se comporta nuestro algoritmo es *Sin síntomas ni signos*. De nuevo, teniendo en cuenta el desglose detallado en la tabla 5.12, observamos que hay 5 pacientes "mal clasificados" en el clúster con elementos que pertenecen mayoritariamente a la clase *EC* y hay 37 pacientes "mal clasificados" en el clúster con elementos que pertenecen mayoritariamente a la clase *no EC*. Por lo tanto, este algoritmo tiene un porcentaje de fallo del 6,86 %.

| Dataset | Silhouette | valor-F | Precisión | Exhaustividad |
|---|---|---|---|---|
| Original | 0.092 | 0.578 | 0.465 | 0.762 |
| Sin país, sexo ni edad | 0.103 | 0.534 | 0.428 | 0.708 |
| Sin síntomas ni signos | 0.122 | 0.572 | 0.459 | 0.759 |
| Sin país, sexo, edad, síntomas ni signos | 0.042 | 0.402 | 0.299 | 0.611 |
| Elementos diagnósticos principales | 0.057 | 0.401 | 0.314 | 0.556 |
| Selección de atributos CFS | 0.081 | 0.53 | 0.419 | 0.722 |
| Selección de atributos FCBF | 0.1 | 0.528 | 0.409 | 0.742 |

Cuadro 5.5: Resultados de los diferentes datasets con el algoritmos $K$-Modes

### 5.3.3. K-Modes

Tras la ejecución del algoritmo sobre cada uno de los datasets, los valores de las métricas de evaluación se pueden ver en la tabla 5.5. A la vista de los valores de las métricas consideramos que el dataset con el que mejor se comporta nuestro algoritmo es *Originial*. Teniendo en cuenta la tabla 5.12, observamos que hay 39 pacientes "mal clasificados" en el clúster con elementos que pertenecen mayoritariamente a la clase *EC* y hay 20 pacientes "mal clasificados" en el clúster con elementos que pertenecen mayoritariamente a la clase *no EC*. El porcentaje de fallo de este algoritmo es del 9,6 %.

### 5.3.4. K-POD

Los valores correspondientes a las métricas de evaluación para cada dataset se pueden ver en la tabla 5.6. A la vista de los valores de las métricas consideramos que el dataset con el que mejor se comporta nuestro algoritmo es *Selección de atributos CFS*. A la vista del desglose detallado de los clústeres presentado en la tabla 5.12, observamos que hay 10 pacientes "mal clasificados" en el clúster con elementos que pertenecen mayoritariamente a la clase *EC* y hay 64 pacientes "mal clasificados" en el clúster con elementos que pertenecen mayoritariamente a la clase *no EC*. Por lo tanto este algoritmo tiene un porcentaje de fallo del 12 %.

### 5.3.5. K-Medoids

Los resultados correspondientes a los diferentes datasets pueden encontrarse en la tabla 5.7. A la vista de los valores de las métricas consideramos que el dataset con el que mejor se comporta nuestro algoritmo es *Sin síntomas ni signos*. En el desglose de los clústeres que aparece en la tabla 5.12 podemos observar que hay 33 pacientes "mal clasificados" en el clúster con elementos que pertenecen mayoritariamente a la clase *EC* y hay 22 pacientes "mal clasificados" en el clúster con elementos que pertenecen mayoritariamente a la clase *no EC*. Por lo tanto este algoritmo tiene un porcentaje de fallo del $8,98\,\%$.

| Dataset | Silhouette | valor-F | Precisión | Exhaustividad |
|---|---|---|---|---|
| Original | 0.072 | 0.457 | 0.32 | 0.8 |
| Acortado | 0.091 | 0.446 | 0.319 | 0.743 |
| Sin país, sexo ni edad | 0.118 | 0.509 | 0.382 | 0.765 |
| Sin país, sexo ni edad acortado | 0.15 | 0.494 | 0.379 | 0.707 |
| Sin síntomas ni signos | 0.08 | 0.578 | 0.46 | 0.776 |
| Sin síntomas ni signos acortado | 0.095 | 0.533 | 0.428 | 0.707 |
| Sin país, sexo, edad, síntomas ni signos | 0.204 | 0.573 | 0.459 | 0.761 |
| Sin país, sexo, edad, síntomas ni signos acortado | 0.188 | 0.474 | 0.373 | 0.651 |
| Elementos diagnósticos principales | 0.118 | 0.536 | 0.418 | 0.745 |
| Selección atributos CFS | 0.26 | 0.572 | 0.458 | 0.759 |
| Selección atributos FCBF | 0.505 | 0.538 | 0.42 | 0.749 |

Cuadro 5.6: Resultados de los diferentes datasets con el algoritmos $K$-POD

| Dataset | Silhouette | valor-F | Precisión | Exhaustividad |
|---|---|---|---|---|
| Original | 0.052 | 0.499 | 0.401 | 0.662 |
| Acortado | 0.023 | 0.422 | 0.338 | 0.562 |
| Sin país, sexo ni edad | -0.21 | 0.46 | 0.299 | 0.997 |
| Sin país, sexo ni edad acortado | 0.016 | 0.452 | 0.344 | 0.656 |
| Sin síntomas ni signos | 0.105 | 0.576 | 0.463 | 0.762 |
| Sin síntomas ni signos acortado | 0.066 | 0.545 | 0.436 | 0.724 |
| Sin país, sexo, edad, síntomas ni signos | -0.299 | 0.46 | 0.299 | 0.997 |
| Sin país, sexo, edad, síntomas ni signos acortado | 0.018 | 0.381 | 0.301 | 0.518 |
| Elementos diagnósticos principales | -1 | 0.458 | 0.297 | 1 |
| Selección atributos CFS | 0.69 | 0.381 | 0.298 | 0.527 |
| Selección atributos FCBF | -1 | 0.458 | 0.297 | 1 |

Cuadro 5.7: Resultados de los diferentes datasets con el algoritmos KMedoids

### 5.3.6. Espectral

Tras la ejecución del algoritmo sobre cada uno de los datasets, los valores de las métricas de evaluación se pueden ver en la tabla 5.8. A la vista de los valores de las métricas consideramos que el dataset con el que mejor se comporta nuestro algoritmo es *Original*. Teniendo en cuenta la tabla 5.13, podemos observar que 28 pacientes están "mal clasificados" en el clúster con elementos que pertenecen mayoritariamente a la clase *EC* y que hay 25 pacientes "mal clasificados" en el clúster con elementos que pertenecen mayoritariamente a la clase *no EC*. Por lo tanto este algoritmo tiene un porcentaje de fallo del 8,6 %.

| Dataset | Silhouette | valor-F | Precisión | Exhaustividad |
|---|---|---|---|---|
| Original | 0.123 | 0.593 | 0.478 | 0.781 |
| Acortado | 0.113 | 0.408 | 0.303 | 0.627 |
| Sin país, sexo ni edad | 0.142 | 0.563 | 0.452 | 0.746 |
| Sin país, sexo ni edad acortado | 0.24 | 0.472 | 0.343 | 0.757 |
| Sin síntomas ni signos | 0.143 | 0.592 | 0.477 | 0.783 |
| Sin síntomas ni signos acortado | 0.102 | 0.406 | 0.305 | 0.606 |
| Sin país, sexo, edad, síntomas ni signos | 0.287 | 0.43 | 0.316 | 0.672 |
| Sin país, sexo, edad, síntomas ni signos acortado | 0.219 | 0.535 | 0.418 | 0.745 |
| Elementos diagnósticos principales | 0.346 | 0.43 | 0.321 | 0.652 |
| Selección atributos CFS | 0.398 | 0.252 | 0.571 | 0.162 |
| Selección atributos FCBF | 0.413 | 0.224 | 0.544 | 0.141 |

Cuadro 5.8: Resultados de los diferentes datasets con el algoritmo de clustering Espectral

| Dataset | Silhouette | valor-F | Precisión | Exhaustividad |
|---|---|---|---|---|
| Original | 0.122 | 0.608 | 0.491 | 0.798 |
| Acortado | 0.115 | 0.403 | 0.304 | 0.597 |
| Sin país, sexo ni edad | 0.135 | 0.567 | 0.453 | 0.756 |
| Sin país, sexo ni edad acortado | 0.136 | 0.529 | 0.41 | 0.744 |
| Sin síntomas ni signos | 0.141 | 0.596 | 0.48 | 0.786 |
| Sin síntomas ni signos acortado | 0.128 | 0.569 | 0.456 | 0.755 |
| Sin país, sexo, edad, síntomas ni signos | 0.159 | 0.59 | 0.475 | 0.78 |
| Sin país, sexo, edad, síntomas ni signos acortado | 0.216 | 0.562 | 0.446 | 0.758 |
| Elementos diagnósticos principales | 0.176 | 0.523 | 0.42 | 0.692 |
| Selección atributos CFS | 0.339 | 0.512 | 0.41 | 0.68 |
| Selección atributos FCBF | 0.48 | 0.528 | 0.409 | 0.742 |

Cuadro 5.9: Resultados de los diferentes datasets con el algoritmo Aglomerativo

### 5.3.7. Aglomerativo

Tras la ejecución del algoritmo, obtuvimos los resultados presentados en la tabla 5.9. A la vista de los valores de las métricas consideramos que el dataset con el que mejor se comporta nuestro algoritmo es con el *Original*. Podemos ver el desglose detallado de los clústeres en la tabla 5.13, observando que hay 25 pacientes "mal clasificados" en el clúster con elementos que pertenecen mayoritariamente a la clase *EC* y que hay 19 pacientes "mal clasificados" en el clúster con elementos que pertenecen mayoritariamente a la clase *no EC*. Por lo tanto este algoritmo tiene un porcentaje de fallo del 7,18 %.

| Dataset | Silhouette | valor-F | Precisión | Exhaustividad |
|---|---|---|---|---|
| Original | -0.173 | 0.499 | 0.346 | 0.896 |
| Acortado | -0.199 | 0.506 | 0.357 | 0.866 |
| Sin país, sexo ni edad | -0.184 | 0.524 | 0.393 | 0.785 |
| Sin país, sexo ni edad acortado | -0.152 | 0.518 | 0.424 | 0.667 |
| Sin síntomas ni signos | -0.168 | 0.506 | 0.356 | 0.872 |
| Sin síntomas ni signos acortado | -0.177 | 0.51 | 0.388 | 0.747 |
| Sin país, sexo, edad, síntomas ni signos | -0.058 | 0.52 | 0.491 | 0.553 |
| Sin país, sexo, edad, síntomas ni signos acortado | 0.19 | 0.279 | 0.559 | 0.186 |
| Elementos diagnósticos principales | 0.058 | 0.487 | 0.505 | 0.47 |
| Selección atributos CFS | 0.594 | 0.189 | 0.637 | 0.111 |
| Selección atributos FCBF | 0.771 | 0.485 | 0.511 | 0.461 |

Cuadro 5.10: Resultados de los diferentes datasets con el algoritmo DBSCAN.

### 5.3.8. DBSCAN

Los resultados correspondientes a este algoritmo aparecen en la tabla 5.10. Es importante mencionar que en este tipo de algoritmos no se puede introducir de antemano el número final de clústeres, lo que hace que sea mucho más difícil establecer un resultado buscado. Hemos variado los diferentes parámetros para obtener el mejor coeficiente de Silhoutte y el mejor valor-F posible mientras intentamos minimizar el número final de clústeres. Sin embargo, nos ha sido imposible obtener un número pequeño de clústeres y, por este motivo, los resultados de este algoritmo no serán mostrados en la tabla final. Por lo tanto, podemos afirmar que aunque los resultados de las métricas parezcan buenos (como por ejemplo para el dataset de *Selección de atributos FCBF*), el resultado de la agrupación no es interpretable pues tenemos un gran número de clústeres heterogéneos y de tamaños muy dispares, incluyendo algunos de ellos un solo paciente. Por este motivo no tendremos en cuenta este algoritmo y no mostraremos sus resultados detallados en la tabla 5.13.

### 5.3.9. OPTICS

Los resultados obtenidos con este algoritmo son realmente malos, véase tabla 5.11, ya que todos los datasets menos uno tienen el coeficiente de Silhouette negativo. Además, como este algoritmo pertenece a la misma familia que el anterior, aquí tampoco se pueden introducir de antemano el número final de clústeres, lo que hace que sea mucho más difícil encontrar una agrupación interpretable. Hemos variado los diferentes parámetros para obtener el mejor coeficiente de Silhoutte y el mejor valor-F posible mientras intentamos minimizar el número final de clústeres. Sin embargo, nos ha sido imposible obtener un número pequeño de clústeres y por este motivo los resultados de este algoritmo no serán mostrados en la tabla final.

| Dataset | Silhouette | valor-F | Precisión | Exhaustividad |
|---|---|---|---|---|
| Original | -0.145 | 0.495 | 0.364 | 0.772 |
| Acortado | -0.124 | 0.489 | 0.345 | 0.835 |
| Sin país, sexo ni edad | -0.192 | 0.496 | 0.372 | 0.745 |
| Sin país, sexo ni edad acortado | -0.189 | 0.475 | 0.358 | 0.708 |
| Sin síntomas ni signos | -0.163 | 0.499 | 0.366 | 0.783 |
| Sin síntomas ni signos acortado | -0.213 | 0.481 | 0.341 | 0.819 |
| Sin país, sexo, edad, síntomas ni signos | -0.346 | 0.469 | 0.432 | 0.513 |
| Sin país, sexo, edad, síntomas ni signos acortado | -0.247 | 0.403 | 0.427 | 0.381 |
| Elementos diagnósticos principales | -0.395 | 0.472 | 0.375 | 0.639 |
| Selección atributos CFS | -0.012 | 0.316 | 0.482 | 0.235 |
| Selección atributos FCBF | 0.04 | 0.326 | 0.442 | 0.258 |

Cuadro 5.11: Resultados de los diferentes datasets con el algoritmo OPTICS.

### 5.3.10. Comparación de la distribución en clústeres

En las tablas 5.12 y 5.13 podemos ver como se distribuyen en los clústeres los pacientes según su diagnóstico. El Clúster 1 se corresponde con aquel que contenga mayoritariamente pacientes de la clase *EC* y el Clúster 2 es el que contiene mayoritariamente pacientes de la clase *no EC*. Además en la siguiente tabla podemos ver el porcentaje de fallo de cada algoritmo con el dataset que mejor se comporta.

| KMeans | KPrototypes | Kodes | KPOD | KMedoids | Espectral | Aglomerativo |
|---|---|---|---|---|---|---|
| 7,2 % | 6,86 % | 9,6 % | 12 % | 8,98 % | 8,6 % | 7,18 % |

Cuadro 5.14: Porcentajes de fallo de los algoritmos que mejor se comportan

A la vista de estos datos es inmediato determinar que el algoritmo que mejor se comporta es el $K$-Prototypes con el dataset *Sin síntomas ni signos*.

| | | KMeans | KPrototypes | KModes | KPod | KMedoids |
|---|---|---|---|---|---|---|
| **Cluster 1** | Aún en estudio | 9 | 4 | 10 | 3 | 11 |
| | EC | 236 | 227 | 242 | 200 | 240 |
| | EC Potencial | 10 | 9 | 13 | 11 | 15 |
| | EC Refractaria | 4 | 3 | 5 | 3 | 5 |
| | EC Dudosa | 6 | 4 | 8 | 2 | 9 |
| | Paciente perdido | 1 | 0 | 1 | 1 | 1 |
| | SGNC no estricta | 3 | 1 | 8 | 1 | 4 |
| | SGNC estricta | 1 | 0 | 2 | 0 | 1 |
| | Sin diagnóstico | 9 | 5 | 13 | 4 | 14 |
| | no EC | 14 | 4 | 29 | 9 | 28 |
| **Cluster 2** | Aún en estudio | 12 | 17 | 11 | 18 | 10 |
| | EC | 26 | 35 | 20 | 62 | 22 |
| | EC Potencial | 27 | 28 | 24 | 26 | 22 |
| | EC Refractaria | 1 | 2 | 0 | 2 | 0 |
| | EC Dudosa | 6 | 8 | 4 | 10 | 3 |
| | Paciente perdido | 0 | 1 | 0 | 0 | 0 |
| | SGNC no estricta | 19 | 21 | 14 | 21 | 18 |
| | SGNC estricta | 3 | 4 | 2 | 4 | 3 |
| | Sin diagnóstico | 80 | 84 | 76 | 85 | 75 |
| | no EC | 145 | 155 | 130 | 150 | 131 |

Cuadro 5.12: Resumen 1 de los resultados de cada algoritmo

| | | Espectral | Agglomerative | DBSCAN | OPTICS |
|---|---|---|---|---|---|
| **Cluster 1** | Aún en estudio | 8 | 11 | - | - |
| | EC | 237 | 243 | - | - |
| | EC Potencial | 12 | 11 | - | - |
| | EC Refractaria | 5 | 5 | - | - |
| | EC Dudosa | 8 | 8 | - | - |
| | Paciente perdido | 1 | 1 | - | - |
| | SGNC no estricta | 6 | 5 | - | - |
| | SGNC estricta | 1 | 1 | - | - |
| | Sin diagnóstico | 9 | 12 | - | - |
| | no EC | 21 | 19 | - | - |
| **Cluster 2** | Aún en estudio | 13 | 10 | - | - |
| | EC | 25 | 19 | - | - |
| | EC Potencial | 25 | 26 | - | - |
| | EC Refractaria | 0 | 0 | - | - |
| | EC Dudosa | 4 | 4 | - | - |
| | Paciente perdido | 0 | 0 | - | - |
| | SGNC no estricta | 16 | 17 | - | - |
| | SGNC estricta | 3 | 3 | - | - |
| | Sin diagnóstico | 80 | 77 | - | - |
| | no EC | 138 | 140 | - | - |

Cuadro 5.13: Resumen 2 de los resultados de cada algoritmo

# Capítulo 6

# Contribuciones individuales al proyecto

## 6.1. Contribuciones de Carla Martínez Nieto-Márquez

Como se ha explicado a lo largo del proyecto, uno de los principales retos que nos encontramos fue el preprocesamiento de los datos y en ello invertimos gran parte de nuestro tiempo. Para poder hacerlo de forma correcta, ambos estuvimos en la reunión con la médica en la que nos explicó el significado de las diferentes campos. Una vez que lo conocimos, nos dividimos el preprocesamiento de los datos ya que cada grupo de columnas tenía un tratamiento diferente debido a su naturaleza. En algunas trabajamos conjuntamente y luego cada uno se encargó de otras de forma individual. En mi caso me encargué de las formatear y tratar las columnas:

- Relacionadas con la genética, es decir, las columnas *HLA:grupos de riesgo, Haplotipo1 y Haplotipo2*.
- Relacionadas con las biopsicas en DSG y en DCG, es decir, las columnas: *Fecha DCG Biopsia1, Fecha DCG Biopsia2, FECHA LIEs DCG_1, FECHA LIEs DCG_2, Fecha DSG biopsia1, Fecha DSG biopsia2, FECHA LIEs DSG_1, FECHA LIEs DSG_2, FECHA LIEs DSG_3, DCG Biopsia-AP1, DCG Biopsia-AP2, AP Biopsia DCG LIEs_1, AP en Biopsia DCG LIEs_2, DSG Biopsia AP1, DSG Biopsia AP2, AP Biopsia DSG LIEs_1, AP en Biopsia DSG LIEs_2, AP en Biopsia DSG LIEs_3*.
- Relacionadas con la prueba *LIEs* tanto en DCG como en DSG, es decir, las columnas: *LIEs DCG %GD_1, LIEs DCG %iNK_1, LIEs DCG %GD_2, LIEs DCG % iNK_2, Valoración DCG LIEs1, Valoración LIEs2, LIEs DSG %GD_1 , LIEs DSG %iNK_1, LIEs DSG %GD_2, LIEs DSG % iNK_2, LIEs DSG %GD_3, LIEs DSG %iNK_3, Valoración DSG LIEs1, Valoración DSG LIEs2, Valoración DSG LIEs3*.

También me encargué de escribir el tratamiento de las nombradas columnas, que se puede encontrar en la sección 2.1.

A continuación llevé a cabo todas las tareas de procesamiento de los datos, es decir, estudié las formas en las que se lleva a cabo tradicionalmente el tratamiento de los datos

categóricos, decidí cuál era la mejor para nuestro tipo de datos y la apliqué a nuestras columnas categóricas, reflejando el proceso en la sección 2.2. También me informé sobre el tratamiento que se da a los datos incompletos. Este estudio lo detallé en la sección 2.3, en un primer momento traté de aplicar el enfoque más utilizado (eliminación) pero quedó totalmente descartado porque se eliminaba prácticamente toda la información relevante del dataset (las pruebas concernientes a este enfoque están detalladas en la sección 2.3.1). Después, estudié los distintos tipos de imputación y decidí cuáles eran los mejores métodos para aplicar, teniendo en cuenta el tipo de datos que con los que trabajábamos. A continuación probé a aplicar sobre nuestro dataset la imputación multivariada (que quedó descartada porque no convergía) y la imputación por los vecinos más cercanos. Todo esto está escrito detalladamente en la sección 2.3.2. Finalmente analicé los distintos métodos para el escalado de los datos, apliqué todos ellos al dataset y decidí cual era el mejor, reflejando el proceso en la sección 2.4.

Mi compañero Pablo y yo tuvimos la oportunidad de acudir al *VII Congreso Nacional de la Sociedad Española de Enfermedad Celíaca* celebrado los días 13 y 14 de mayo. Este congreso consistió en acudir a diversas conferencias virtuales impartidas por profesionales de la enfermedad celíaca. Tras acudir a dichas conferencias, adquirimos un mayor conocimiento sobre la enfermedad celíaca y por ello consideré interesante generar un dataset más que contuviera únicamente las columnas más relevantes en el diagnóstico de la celiaquía. Además, tuvimos una reunión con Manuel Méndez, un estudiante del grupo de investigación de Mercedes Merayo que dispone conocimientos de Machine Learning el cual nos recomendó que aplicáramos técnicas de selección de variables que fueran fácilmente interpretables como CFS (Correlation based Feature Selection) y FCBF (Fast Correlation-Base Filter). Mi trabajó consistió en buscar una implementación en Python de estos métodos (pues no están en la librería Sklearn) y aplicarlos sobre nuestro dataset. Además por ser yo la persona que iba a trabajar con algoritmos que utilizaban datasets distintos al numérico con todos los datos completos generé los datasets categoricos, mixto e incompleto iniciales (es decir, con todas las columnas). La descripción de estos datasets se puede encontrar en la sección 2.5.

Tras el procesamiento de los datos me centré en el estudio teórico y la aplicación sobre nuestros datasets de algunos algoritmos. Comencé estudiando el algoritmo *K-means*, cuya base teórica reflejé en la sección 3.1.1. Después, investigué sobre algoritmos que se pudieran aplicar sobre datasets que incluyeran datos mixtos (es decir, sobre datasets que contuvieran tanto datos categóricos como datos numéricos) de manera que pudiéramos trabajar directamente con las columnas categóricas y no tuviéramos que hacer la transformación a columnas binarias. En esta búsqueda encontré el algoritmo *K-Prototypes*, cuya base teórica escribí en la sección 3.1.2, Sin embargo, este algoritmo no se encuentra implementado en la librería Sklearn de Python, que es la que hemos utilizado para aplicar gran parte de los algoritmos de clustering testeados en este trabajo. Por este motivo, busqué una implementación fiable en Python del mismo. En el repositorio de Github donde encontré la implementación de este algoritmo estaba también implementado el algoritmo *K-Modes* que trabaja únicamente con datos categóricos. Puesto que toda la información de las columnas numéricas está recogida también en alguna columna categórica, consideré que podía tener sentido trabajar únicamente con datos categóricos. El estudio teórico de este método está descrito en la sección

3.1.3. Dada la naturaleza de nuestros datos y la gran cantidad de datos incompletos, consideré interesante aplicar sobre el dataset un algoritmo especialmente diseñado para tratar con datasets con datos incompletos y estudiar si este se comportaba mejor que los algoritmos que trabajan con el dataset completo (y por tanto con datos imputados). En mi búsqueda encontré el algoritmo *K-POD*, cuyo estudio teórico describí en la sección 3.1.4. Y como en el caso de los dos algoritmos anteriores la implementación de este no estaba en la librería Sklearn de Python, por lo que tuve que buscar una implementación fiable del mismo. Finalmente, utilicé un algoritmo ampliamente conocido que es el de clustering espectral, cuya base teórica reflejé en la sección 3.3.

Como ya he dicho además del estudio teórico de los nombrados algoritmos me encargué de su aplicación a nuestros datasets.

Todos los métodos anteriores necesitan recibir el número de clústeres como parámetro y una forma muy habitual de seleccionarlo es utilizar el coeficiente de Silhouette. Por eso llevé a cabo el estudio teórico del coeficiente que se puede encontrar en 4.2.1 y lo apliqué a los algoritmos anteriores para determinar el mejor número de clústeres.

Una vez mi compañero y yo habíamos aplicado los distintos métodos de clustering sobre los distintos datasets, surgieron de manera natural dos preguntas ¿Son los datasets que hemos definido lo suficientemente buenos? ¿Qué métodos se comportan mejor? Para dar respuesta a estas dos preguntas estudié el estadístico de Hopkins (cuya base teórica escribí en 4.1) y lo apliqué sobre nuestros datasets, para lo cual fue necesario que buscara una implementación en Python del mismo. También me informé sobre los distintos métodos de análisis de los resultados de clustering (extrínsecos e intrínsecos) y seleccioné uno de cada (*BCubed* como método extrínseco y *coeficiente de Silhouette* como método intrínseco) y detallé en las secciones 4.2.2 y 4.2.1 sus descripciones teóricas. Además, busqué una implementación del primero en Python (pues no está en la librería *Sklearn*) y apliqué ambos sobre los resultados de todos los algoritmos.

También, me encargué de escribir los resultados de la evaluación de los datasets y de los algoritmos que estudié teóricamente e implementé, es decir, escribí las secciones 5.2, 5.3.1, 5.3.2, 5.3.3, 5.3.4, 5.3.6. Además llevé a cabo los experimentos previos que reflejé en las secciones 5.1.1 y 5.1.1.

A continuación, me encargué de la generación de los elementos gráficos, que se generan cuando se ejecuta nuestro programa. Por cada algoritmo y dataset se muestra la gráfica con los valores del *coeficiente de Silhouette* para cada número de clústeres del 2 al 20, una herramienta interactiva en $2D$ que representa los elementos del dataset con sus datos y los clústeres a los que se han asignado y una tabla resumen de cuántos pacientes hay por cada tipo de diagnóstico en cada uno de los clústeres. Además, se genera una tabla resumen de cada uno de los algoritmos, donde se comparan los valores para el coeficiente de *Silhouette*, y para la precisión y exhaustividad *BCubed*. Se puede encontrar una pequeña muestra de estos elementos gráficos en el apéndice B.

También traduje el capítulo 1.

Finalmente escribí la documentación del repositorio de Github, es decir escribí el Readme de la página y me aseguré de que la ejecución en local funcionara correctamente. También escribí el apéndice A donde expliqué más detalladamente la ejecución en local del código y

la estructura del repositorio.

## 6.2. Contribuciones de Pablo Sanz Caperote

Como ya ha explicado mi compañera, una gran parte de nuestro trabajo se centró en el formateo de los datos y este fue nuestro punto de partida. Tras haber asistido a la reunión con la médico y haberme informado brevemente sobre los aspectos básicos de la enfermedad celíaca, me encargué de formatear y tratar las siguientes columnas:

- *Indique país de origen o en su defecto la información disponible.*
- *Edad diagnóstico.*
- Las columnas referentes al grado de parentesco: : *Grupo de riesgo*, *Especifique el familiar/es afectado/s*, *Grado de parentesco*, *Grado de parentesco (si hay más de 1).*
- Las columnas referentes a las enfermedades inmunológicas: *Enfermedad inmunológica*, *Enfermedad inmunológica (si hay más de 1)*, *Enfermedad inmunológica (si hay más de 2)*, *Otro/s riesgo/s.*
- Las columnas referentes a los síntomas y signos: *Síntomas específicos*, *Síntomas específicos*, *Síntomas específicos*, *Otros síntomas*, *Signos*, *Signos 2*, *Signos 3.*
- Las columnas referentes a los anticuerpos ATG2 y EMA tanto en DCG como en DSG: *DCG ATG2_1*, *Indicar titulo del anticuerpo (DCG ATG2_1)*, *Indicar el kit empleado con el punto de corte entre paréntesis*, *Otros kits*, *DCG EMA*, *Indicar el kit empleado con el punto de corte entre paréntesis*, *DCG ATG2_2*, *Indique título de anticuerpo (DCG ATG2_2)*, *DSG ATG2_1*, *Indique el título del anticuerpo (DSG ATG2_1)*, *Indicar el kit empleado con el punto de corte entre paréntesis* , *Otros Kits*, *DSG EMA*, *DSG ATG2_2*, *Indique el título del anticuerpo (DSG ATG2_2).*

También me encargué de escribir el tratamiento de las nombradas columnas, que se puede encontrar en la sección 2.1.

Una vez que ya teníamos los datos procesados, llegó el momento de seleccionar algoritmos de clustering y estudiar su funcionamiento. Mi tarea aquí fue tener un conocimiento general de diferentes métodos de clustering para saber qué tipos existen y en qué se diferencian unos de otros. De esta forma, pudimos seleccionar algoritmos de cada una de las diferentes familias existentes para así ocupar todo el espectro de métodos, ya que al tener una base de datos tan especial; no podíamos saber con anterioridad qué familia de algoritmos nos iba a proporcionar mejores resultados. Con todo esto, escribí la introducción de las secciones 3 y 3.1.
Además, me encargué concretamente del análisis teórico de varios algoritmos. El primero de ellos fue el algoritmo *KMedoids*, porque aunque ya teníamos varios algoritmos de la familia de algoritmos de partición, me creía que este nuevo enfoque podía generar podía generar

resultados interesantes dada la naturaleza del dataset (sección 3.1.5). El siguiente algoritmo que exploré fue el *Cluster aglomerativo*, ya que me pareció de utilidad este nuevo tipo de algoritmos que son tan usados (sección 3.2). Por último, estudié la familia de algoritmos de densidad en la que utilicé los métodos *DBSCAN* y *OPTICS* (sección 3.4.1 y 3.4.2). Además de su análisis teórico, me encargué de la aplicación de dichos algoritmos a nuestros diferentes datasets. Sin embargo, la elección de parámetros en estos algoritmos no fue tan sencilla como en el resto, ya que el único de los cuatro en el que se puede determinar el número final de clústeres es en el algoritmo KMedoids. Los otros tres algoritmos, tienen diferentes parámetros que determinan el resultado final de una forma no tan trivial. Para conseguirlo realicé un gran número de pruebas con diferentes combinaciones hasta lograr una combinación que producía unos buenos resultados.

Otra tarea que he llevado a cabo ha sido la búsqueda de mejores resultados cambiando los datasets. En un primer momento, teníamos únicamente un dataset con las columnas que nos había dicho la médico. Mi labor aquí, fue analizar las columnas y crear nuevos datasets eliminando algunas que consideraba prescindibles después de haber acudido al *VII Congreso Nacional de la Sociedad Española de la Enfermedad Celíaca*. De esta forma, creé diferentes combinaciones del conjunto inicial con el objetivo de reducir la complejidad del mismo formando los datasets *Dataset numérico 3, Dataset numérico 4, Dataset numérico 5, Dataset numérico 6, Dataset numérico 7* y *Dataset numérico 8* (sección 2.5).

Llegados a este punto, ya estábamos en condiciones de analizar los diferentes resultados. Y por seguir con la misma línea de trabajo que aplicamos para la sección teórica, me encargué de analizar los resultados de los algoritmos *KMedoids*, *Cluster aglomerativo*, *OPTICS* y *DBSCAN*. En cada uno de ellos, tuve que seleccionar el dataset que establecía una mejor agrupación de los datos basándome en las métricas que tenemos y viendo la composición de detallada de cada grupo resultante. Después de esto, se contaba para cada uno de ellos los resultados obtenidos tratando de explicar el porqué de cada uno de ellos tanto los resultados malos como los buenos.

Para terminar con la memoria del trabajo, me encargué de definir la introducción, el estilo gráfico de las tablas, las conclusiones y una línea de trabajo futuro que creemos que puede servir para continuar con el proyecto que nosotros hemos comenzado y conseguir de esta forma unos mejores resultados.

Por último, para poder ayudar a las futuras personas que continúen con este proyecto y utilicen nuestros archivos, creé un archivo en nuestro repositorio de Github que simulara los datos médicos de los pacientes, ya que los datos reales no pueden ser compartidos por la Ley de Protección de Datos. En un primer momento, cree un archivo resumen que recogía las columnas más importantes para diagnosticar la enfermedad celíaca, añadiendo en la última columna el diagnóstico del paciente. No obstante, este archivo fue modificado finalmente para hacer más sencillo el proceso de prueba por si alguien quiere probar todo nuestro proyecto. Así que finalmente, el archivo contiene las mismas columnas y que el archivo

original proporcionado por la médico, únicamente cambiando los datos para no infringir la ley.

# Capítulo 7

# Conclusión y trabajo futuro

Aunque los resultados finales vistos en el capítulo 5 no nos permitan alcanzar con rotundidad uno de los objetivos iniciales del proyecto, el de obtener grupos claramente diferenciados entre pacientes con el mismo diagnóstico, sí que podemos diferenciar entre pacientes celíacos y no celíacos con cierto grado de seguridad. Además, después de todo nuestro trabajo hemos podido sacar algunas conclusiones valiosas.

## 7.1. Calidad de los datos

Como hemos manifestado a lo largo de este trabajo, la calidad de los datos tiene una gran importancia a la hora de aplicar un algoritmo de *Machine Learning*. En nuestro caso, la calidad de los datos era muy baja, pues teníamos una gran porcentaje de casillas vacías y un número bastante limitado de filas (en torno a 600). A pesar de haber aplicado numerosas técnicas de transformación del conjunto de datos como la imputación, el escalado y la transformación de columnas categóricas a columnas binarias, no hemos conseguido obtener un dataset que nos permitiera crear clústeres bien diferenciados entre pacientes con el mismo diagnóstico.

También hemos observado que los datasets que obtenían mejores valores para el coeficiente de Hopkins no son los que mejores resultados obtienen en nuestras pruebas, a pesar de que en teoría deberían ser los que formaran clústeres más diferenciados entre sí. Esto se debe a que hemos forzado la formación de 2 clústeres y es muy posible que con estos datasets se obtuvieran en realidad muchos más.

Una línea importante de trabajo futuro sería el mejorar la calidad de la base de datos con la que trabajamos. Sin embargo, sabemos que no es una tarea fácil ya que uno de los principales problemas a nivel médico para el diagnóstico de esta enfermedad es la falta de pruebas. Debido a esto, si no fuera posible obtener una mejor base de datos, un punto importante sería tratar de mejorar la imputación de los valores vacíos ya que esto podría suponer un gran cambio en los resultados.

## 7.2. Métodos de clustering

También podemos sacar importantes conclusiones sobre los métodos de clústering. En primer lugar, podemos afirmar que los algoritmos basados en densidad DBSCAN y OPTICS no son buenos para este proyecto. Ambos reflejan unos resultados pésimos en general y mucho peores si los comparamos con otros tipos de algoritmos. Para el primero de ellos existe una explicación lógica: al necesitar el algoritmo un parámetro de densidad global, y tener un conjunto de datos tan variado en los que hay algunas zonas de puntos muy densas pero otras muy diluidas, el algoritmo no puede establecer esta clasificación correctamente. Sin embargo, para el segundo de ellos nos resulta más difícil encontrar una explicación ya que debería adaptarse a las diferentes densidades del conjunto de datos. No obstante, suponemos que no consigue realizarlo adecuadamente dada la alta complejidad, debido al gran número de columnas, del dataset.

Además, hemos podido comprobar que de los métodos alternativos al $K$-Means, el $K$-Prototypes es el único que consigue mejorar los resultados del primero, mientras que el $K$-Modes y $K$-POD los empeoran considerablemente. El hecho de que el $K$-POD se comporte peor que el $K$-Means muestra que la imputación que hemos implementado funciona mejor que asignar a cada elemento vacío el valor del centro del clúster para ese campo, que es el procedimiento seguido por el algoritmo $K$-POD. Por otro lado, que el $K$-Modes se comporte peor que los otros algoritmos se puede deber a que trabaja con una cantidad de columnas más pequeña y, por tanto, a los algoritmos parece que les viene bien tener información redundante (recogida en las columnas categóricas y en las numéricas). Por otro lado, los algoritmos de Clustering Espectral y $K$-Medoids obtienen unos resultados suficientemente buenos aunque los que mejor se comportan son el $K$-Means, el $K$-Prototypes y el Clustering Aglomerativo.

Una de las posibles mejoras consiste en no limitarnos a trabajar con 2 clústeres y utilizar el coeficiente de Silhouette para obtener el número óptimo de clústeres y determinar si alguna de las agrupaciones es válida. Para ello, necesariamente deberíamos trabajar mano a mano con un profesional médico pues nosotros, como ya hemos dicho, no somos capaces de interpretar los clústeres obtenidos. De esta forma podríamos tratar de obtener "tipos" de pacientes dentro de aquellos que tengan el mismo diagnóstico y dar así respuesta a uno de los objetivos iniciales de este trabajo. Desgraciadamente, esta no parece una tarea sencilla.

Como última propuesta, siguiendo la línea original del trabajo, se podría tratar de aplicar modelos de clasificación de aprendizaje supervisado en lugar de los métodos de clústering empleados al mismo conjunto de datos inicial y comparar los resultados con los nuestros.

# Capítulo 7

# Conclusions and Future work

Although the final results that have been seen in the chapter 5 do not allow us to reach one of the initial goals of this project: to obtain patterns of patients with the same diagnosis. We are able to differentiate between patients suffering from Celiac Disease and patients that aren't. Moreover we have extracted some interesting conclusions.

## 7.1. Data quality

As you may already know, the quality of the data is the most important factor when successfully applying a Machine Learning algorithm. In our case the data quality was low because we have a lot of missing data and not many rows (600). Despite having applied many transformations to the dataset: scale, imputation and transform categorical columns into binary we haven't been able to obtain a dataset that enables the creation of differentiated clusters with patients with the same diagnosis.

We have also concluded that the dataset with the best Hopkins coefficient values weren't the ones with the best results in our experiments. This is maybe due to the fact that we have forced the creation of 2 clusters and the data in these datasets may be distributed in many clusters.

An important improvement for our project would be to increase the quality of the database we are working with. However, we also know that it will not be an easy task, since one of the most common problems of celiac disease is the lack of medical tests. Another betterment would be to improve the imputation.

## 7.2. Clustering algorithms

We have also extracted some important conclusions regarding clustering algorithms. Firstly we can say that the density algorithms: DBSCAN and OPTICS are not good for this project. Both produce awful results in general, and much worse if we compare them with the others algorithms. DBSCAN results can be explained because it uses a global density parameter and because there are dense areas and others very diluted the algorithm can not

group the data correctly. However, we do not have a logical explanation for OPTICS. We suppose that the high complexity of the dataset might be the problem.

Moreover, among the alternative methods to $K$-Means, $K$-Prototypes is the only one that gets better results than $K$-Means. $K$-Modes and $K$-POD generate worse results. The fact that the algorithm $K$-POD gets worse results than $K$-Means demonstrates that the imputation that we have implemented works better than assigning each empty element the value of the cluster centre which is the procedure followed by $K$-POD. The fact that $K$-Modes obtains worse results than $K$-Means demonstrates that it is interesting to work with redundant information.

Also, Spectral clustering and $K$-Medoids have obtained good enough results, but the best methods are $K$-Means, $K$-Prototypes and Agglomerative clustering.

One of the improvements consists in not limiting the number of clusters to 2 and using the Silhouette coefficient to obtain the optimal number of clusters. For doing so we would need to work hand-by-hand with a health expert so that we can interpret the obtained clusters. By doing we would be able to get "types" of patients within the same diagnosis and we would reach the other initial objective. But this is not an easy task.

Finally, we propose that supervised learning can be used instead of clustering methods.

# Bibliografía


[1] Mihael Ankerst, Markus Breunig, Hans-Peter Kriegel, and Joerg Sander. Optics: Ordering points to identify the clustering structure. *Sigmod Record*, 28(2):49–60, 06 1999.

[2] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. Optics-of: Identifying local outliers. In Jan M. Żytkow and Jan Rauch, editors, *Principles of Data Mining and Knowledge Discovery*, pages 262–270, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[3] Ricardo J. G. B. Campello, Davoud Moulavi, and Joerg Sander. Density-based clustering based on hierarchical density estimates. In Jian Pei, Vincent S. Tseng, Longbing Cao, Hiroshi Motoda, and Guandong Xu, editors, *Advances in Knowledge Discovery and Data Mining*, pages 160–172, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[4] Jocelyn Chi, Eric Chi, and Richard Baraniuk. k -pod a method for k -means clustering of missing data. *The American Statistician*, 70:1–29, 09 2015.

[5] Google Developers. Prepare data. https://developers.google.com/machine-learning/clustering/prepare-data.

[6] J. K. Dixon. Pattern recognition with partly missing data. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(10):617–621, 1979.

[7] Federación de Asociaciones de Celíacos de España. Qué es la enfermedad celiaca. https://celiacos.org/enfermedad-celiaca/que-es-la-enfermedad-celiaca/.

[8] Carlota García-Hoz Garbiñe Roy. Linfograma de linfocitos intraepiteliales por citrometría de flujo: nueva herramienta de diagnóstico. https://celiacos.org/linfograma-de-linfocitos-intraepiteliales-por-citrometria-de-flujo-herramienta-diagnostico-enfermedad-celiaca/#:~:text=LIEs%20y%20enfermedad%20celiaca,lugar%20de%20aceptarlo%20y%20tolerarlo.

[9] Jiawei Han, Micheline Kamber, and Jian Pei. 10 - cluster analysis: Basic concepts and methods. In Jiawei Han, Micheline Kamber, and Jian Pei, editors, *Data Mining (Third Edition)*, The Morgan Kaufmann Series in Data Management Systems, pages 443–495. Morgan Kaufmann, Boston, third edition edition, 2012.

[10] Zhexue Huang. Clustering large data sets with mixed numeric and categorical values. In *In The First Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 21–34, 1997.

[11] Purva Huigol. Feature transformation and scaling techniques to boost your model performance. https://www.analyticsvidhya.com/blog/2020/07/types-of-feature-transformation-and-scaling/.

[12] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[13] Hans-Peter Kriegel, Erich Schubert, and Arthur Zimek. The (black) art of runtime evaluation: Are we comparing algorithms or implementations? *Knowl. Inf. Syst.*, 52(2):341–378, August 2017.

[14] Scikit learn developers. Compare the effect of different scalers on data with outliers. https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html#:~:text=QuantileTransformer%20provides%20non%2Dlinear%20transformations,stabilize%20variance%20and%20minimize%20skewness.

[15] Scikit learn developers. Imputation of missing values. https://scikit-learn.org/stable/modules/impute.html#imputation-of-missing-values.

[16] Ulrike Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, December 2007.

[17] David MacKay. An example inference task: Clustering. *Information Theory, Inference, and Learning Algorithms*, pages 284–292, 2003.

[18] Samuele Mazzanti. Beyond one-hot. 17 ways of transforming categorical features into numeric features. https://towardsdatascience.com/beyond-one-hot-17-ways-of-transforming-categorical-features-into-numeric-features-57f54f199ea4.

[19] Fionn Murtagh and Pedro Contreras. Methods of hierarchical clustering. *CoRR*, abs/1105.0121, 2011.

[20] Hae-Sang Park and Chi-Hyuck Jun. A simple and fast algorithm for k-medoids clustering. *Expert Systems with Applications*, 36(2, Part 2):3336–3341, 2009.

[21] Alberto Rubio-Tapia, Claire L. Jansson-Knodell, Mussarat W. Rahim, Jacalyn A. See, and Joseph A. Murray. Influencia del género en la presentación clínica y enfermedades asociadas en adultos con enfermedad celíaca (ec). *Gaceta Médica de México*, pages 38–46, 2016.

[22] Juan Ignacio Serrano. Diagnóstico. https://www.celiacosmadrid.org/patologias-por-sensibilidad-al-gluten/la-enfermedad-celiaca/diagnostico/.

[23] K. Wagstaff and V. Laidler. Making the most of missing values: Object clustering with partial data in astronomy. *Proceedings of Astronomical Data Analysis Software and Systems XIV*, 347:172, 11 2005.

[24] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1101–1113, 1993.

[25] Concepción Núñez y José Antonio Garrote. Recomendaciones para la elaboración e interpretación de informes genéticos en enfermedad celíaca. *Revista Española de Enfermedades Digestivas*, 110:458 – 461, 00 2018.

[26] Álvaro Noriega Moreno y José Antonio Garrido Sualdea. Herramienta de apoyo al análisis de datos para diagnóstico de enfermedad celiaca, 2020.

# Apéndice A

# Repositorio y ejecución en local

Podemos encontrar el repositorio en Github del proyecto en: https://github.com/TFG-Informatica-Enfermedad-Celiaca/Analisis-EC.

## A.1. Descarga y ejecución en local

Paso 1 **Instala Python 2.7 y Python3.** Puedes instalarlos desde los siguientes links https://www.python.org/downloads/release/python-2716/ y https://www.python.org/downloads/release/python-395/.

Paso 2 **Instala Anaconda y sigue los pasos de instalación.** Puedes instalar Anaconda desde aquí https://www.anaconda.com/products/individual. Después, solo tendrás que seguir los pasos de la instalación y esperar unos minutos a que se complete.

Paso 3 **Instala git bash.** Puedes instalar git bash desde aquí https://git-scm.com/downloads. Después, solo tendrás que seguir los pasos de instalación y esperar unos minutos a que se complete.

Paso 4 **Descarga el repositorio del proyecto en local.** Abre la consola de git bash y navega hasta el directorio donde deseas descargar el proyecto. A continuación solo tendrás que ejecutar

```
git clone https://github.com/TFG-Informatica-Enfermedad-Celiaca/Analisis-EC.git
```

Paso 5 **Descarga el repositorio de la librería scikit-feature.** Desde la consola de git bash, métete dentro del directorio del repositorio anterior y descarga este repositorio. Para ello tienes que ejecutar:

```
git clone https://github.com/jundongl/scikit-feature
```

Paso 6 **Cambios en el repositorio scikit-feature.** Las funciones que utilizaremos tienen algunas erratas que debemos arreglar para el correcto funcionamiento del proyecto. Para ello, en el explorador de archivos iremos al directorio donde hayamos descargado el repositorio de scikit-feature y navegaremos a *skfeature/function/information_theoretical_based/FCBF.py* y abrir el fichero con un editor de texto para cambiar la línea 39 sustituyendo el valor actual ~~t1=np.zeros((n_features, 2), dtypes='object')~~ por t1=np.zeros((n_features, 2), dtype='object') y en la línea 53 tenemos que sustituir ~~fp= X[:, s_list[idx, 0]]~~ por fp= X[:, int(s_list[idx, 0])].

Paso 7 **Instala las librerías necesarias.** Abre Spyder y ejecuta en la consola de Spyder los siguientes comandos [1].

```
conda install numpy
conda install pandas
conda install -c plotly plotly
conda install -c conda-forge scikit-learn
conda install -c conda-forge scikit-learn-extra
pip install kPOD
pip install kmodes
conda install -c conda-forge umap-learn
conda install matplotlib
pip install pyclustertend
conda install -c anaconda scipy
```

Paso 8 **Instala el repositorio scikit-feature.** Abre la consola de tu sistema operativo y navega hasta el directorio donde hayas descargado el repositorio de scikit-feature.

Paso 8.1 Si eres usuario de Windows ejecuta:

```
setup.py install
```

Paso 8.2 Si eres usuario de Linux ejecuta:

```
python setup.py install
```

Paso 9 **Ejecuta fichero source.py.** Abre en Spyder el fichero source.py y ejecútalo. La ejecución tardará unos minutos, cuando termine podrás encontrar los resultados en la carpeta */images*.

Ten en cuenta que los resultados obtenidos tras la ejecución no van a coincidir con los mostrados en el capitulo 5 porque, por protección de datos, el archivo que se incluye en el repositorio es un fichero de prueba con datos que no son los reales.

[1] Si la ejecución de alguno de estos comandos te da un error prueba ejecutándolo con el flag −−user.

## A.2. Estructura del repositorio

En el repositorio encontramos los siguientes archivos:

- *source.py*. Archivo principal donde se llama al preprocesado y a la ejecución de los métodos de clustering. Además, se generan la tabla del estadístico de Hopkins B.2 y la tabla resumen de cada algoritmo B.5.
- *Fichero prueba.xlsx*. Archivo que contiene datos que simulan a los datos reales porque, como ya hemos dicho, por motivos de protección de datos no podemos publicar los datos originales.
- *Important columns.xlsx*. Archivo que contiene los nombres de las columnas que la médica consideraba importantes.
- *preprocessing/load_ data.py*. Fichero necesario para cargar los datos de los ficheros .xlsx.
- *preprocessing/format_ data.py*. En este fichero se implementa todo el formateo inicial de las columnas descrito en la sección 2.1.
- *preprocessing/categorical_ data.py*. Este es un fichero muy breve donde se hace el tratamiento de las columnas categóricas descrito en la sección 2.2.
- *preprocessing/deletion.py*. Este fichero incluye los experimentos de eliminación de filas y columnas descritos en la sección 2.3.1.
- *preprocessing/imputation.py*. Este fichero incluye tanto la imputación multivariada como la imputación por vecinos más cercanos (la versión previa de esta imputación está comentada). Esto corresponde con lo explicado en la sección 2.3.2.
- *preprocessing/scale.py*. Incluye funciones para todos los posibles escalados descritos en la sección 2.4.
- *preprocessing/preprocess.py*. En este fichero se generan los distintos datasets descritos en 2.1.
- *clustering/DBSCAN.py, clustering/agglomerative.py, clustering/kmeans.py, clustering/kmedoids.py, clustering/kmodes_prototypes.py, clustering/kpod_ a.py, clustering/optics.py, clustering/spectral.py*. Ejecución de los algoritmos descritos en las secciones 3.4.1, 3.2, 3.1.1, 3.1.5, 3.1.3, 3.1.2, 3.1.4, 3.4.2, 3.3.
- *clustering/b3.py*. Este fichero incluye exactamente el código del repositorio `https://github.com/m-wiesner/BCUBED`. Copiamos el código, en lugar de clonar el repositorio como hacemos con la librería `scikit-feature`, porque el fichero de instalación no funciona correctamente.

- *clustering/hopkins_statistics.py*. Fichero brevísimo donde llamamos a la función que calcula el coeficiente de Hopkins para un dataset.
- *clustering/rater.py*. Dado un dataset y el resultado de un método de clústering, este genera la tabla B.4.
- *clustering/reduceDimension.py*. Dado un dataset y el resultado de un método de clústering, este genera la gráfica B.3.
- *clustering/silhouette.py*. Calcula el mejor valor para el coeficiente de Silhouette y genera la gráfica B.1.

# Apéndice B

# Elementos gráficos

Una parte fundamental de nuestro trabajo consistió en interpretar y entender los resultados obtenidos y ser capaces de explicárselos a la médica durante nuestras reuniones. Para eso fue esencial la generación de gráficas y tablas que reflejaran resumidamente los resultados. Aquí vamos a explicar muy brevemente las principales gráficas y tablas generadas por nuestro programa. Todas ellas han sido generadas utilizando Plotly pues consideramos que la estética de las gráficas es muy superior al de otras librerías y, además, esta librería permite generar gráficas y tablas interactivas.

## B.1. Coeficiente de Silhouette para un algoritmo y datasets concretos

Se genera una gráfica que contiene para cada número $k$ entre 2 y 20 el valor del coeficiente de Silhouette para un algoritmo y un dataset concreto cuando se utiliza un número de clústeres $k$. Esta gráfica es interactiva y si se pasa el ratón por encima de ella se muestra el valor exacto del coeficiente.

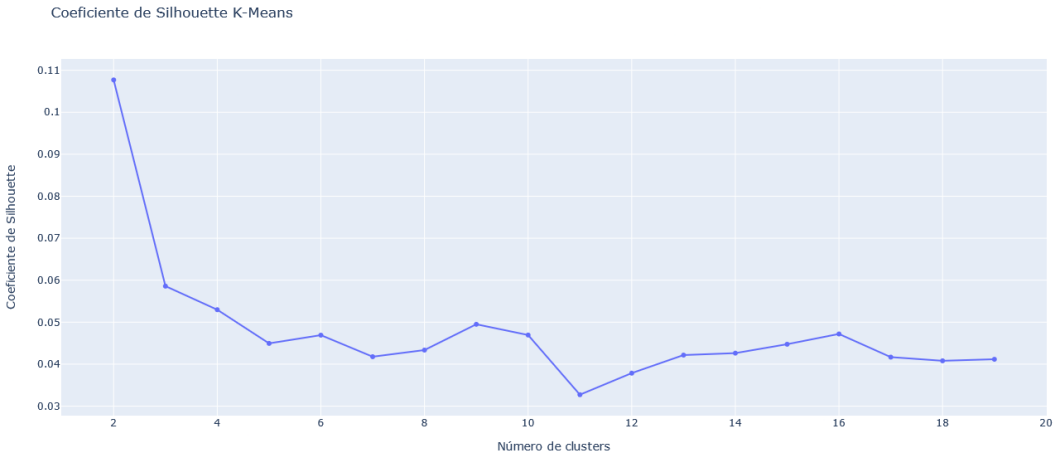
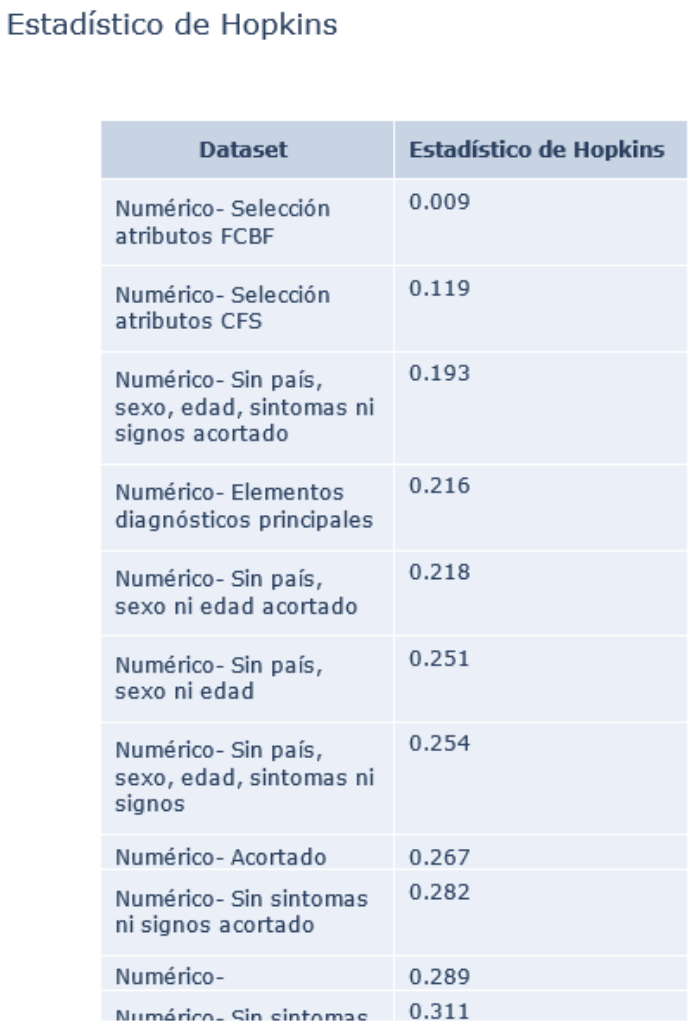



Figura B.1: Coeficiente de Silhouette

## B.2. Tabla estadístico de Hopkins

Se genera una tabla que contiene los valores del estadístico de Hopkins para cada dataset ordenados de menor a mayor.



Estadístico de Hopkins

| Dataset | Estadístico de Hopkins |
|---|---|
| Numérico- Selección atributos FCBF | 0.009 |
| Numérico- Selección atributos CFS | 0.119 |
| Numérico- Sin país, sexo, edad, sintomas ni signos acortado | 0.193 |
| Numérico- Elementos diagnósticos principales | 0.216 |
| Numérico- Sin país, sexo ni edad acortado | 0.218 |
| Numérico- Sin país, sexo ni edad | 0.251 |
| Numérico- Sin país, sexo, edad, sintomas ni signos | 0.254 |
| Numérico- Acortado | 0.267 |
| Numérico- Sin sintomas ni signos acortado | 0.282 |
| Numérico- | 0.289 |
| Numérico- Sin sintomas | 0.311 |

Figura B.2: Valores del estadístico de Hopkins

## B.3. Gráfico interactivo para un algoritmo y datasets concretos

Esta es una gráfica donde se muestra cada paciente de la base de datos como un símbolo distinto en función de su diagnóstico (rombo: paciente con EC, círculo: paciente no EC ni SGNC, cuadrado: paciente con EC potencial...) y con un color distinto en función del clúster al que ha sido asignado.

Esta gráfica es una representación en 2D, por lo que se ha tenido que llevar a cabo una reducción de la dimensionalidad. Además, la gráfica es interactiva y nos permite seleccionar solo los pacientes en los que estemos interesados por diagnóstico o clúster. También, si pasamos el ratón por encima de algún paciente podemos ver información sobre el mismo como, por ejemplo, su identificador, sexo, HLA: grupo de riesgo y algunos resultados de pruebas como ATG2, EMA... En concreto esta es la herramienta que utilizamos cuando le enseñamos a la médica los resultados que habíamos obtenido.

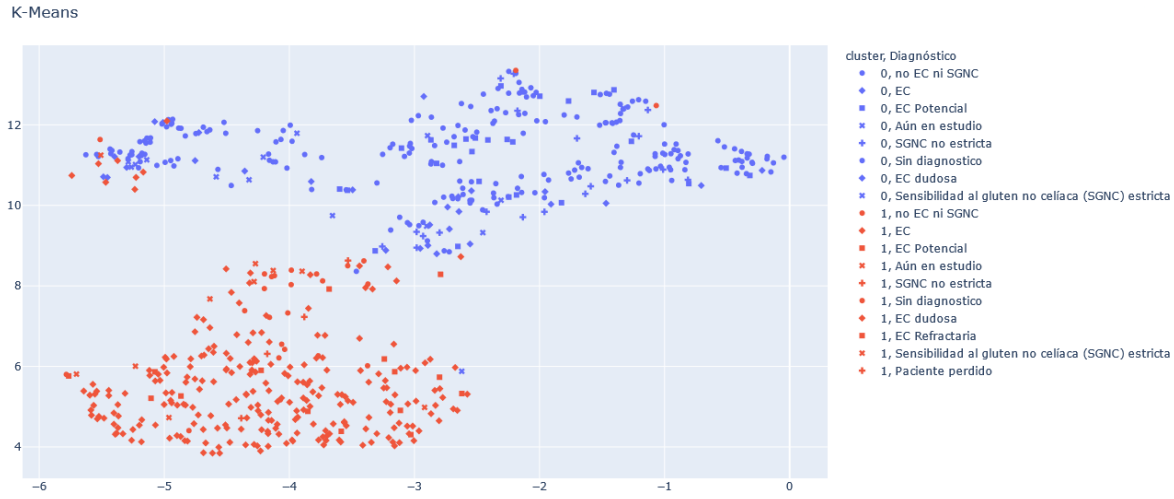



Figura B.3: Gráfico interactivo

## B.4. Tabla clasificatoria de los pacientes

Esta tabla muestra para cada clúster el número de individuos que hay con cada diagnóstico cuando se ejecuta un algoritmo de clustering concreto sobre un dataset determinado. Además, se muestra el mejor coeficiente de Silhouette y los valores para la precisión, exhaustividad y valor-F Bcubed.

K-Means

| Diagnóstico | Número |
|---|---|
| Aún en estudio | 12 |
| EC | 25 |
| EC Potencial | 27 |
| EC dudosa | 4 |
| SGNC no estricta | 19 |
| Sensibilidad al gluten no celíaca (SGNC) estricta | 3 |
| Sin diagnostico | 80 |
| no EC ni SGNC | 143 |
| Aún en estudio | 9 |
| EC | 237 |
| EC Potencial | 10 |
| EC Refractaria | 5 |
| EC dudosa | 8 |
| Paciente perdido | 1 |
| SGNC no estricta | 3 |
| Sensibilidad al gluten no celíaca (SGNC) estricta | 1 |
| Sin diagnostico | 9 |
| no EC ni SGNC | 16 |

| Silhouette | Valor-F | Precisión | Exhaustividad |
|---|---|---|---|
| 0.10770433 | 0.60964406 | 0.49157021 | 0.80237156 |

Figura B.4: Tabla clasificatoria

## B.5. Tabla resumen para un algoritmo

Esta tabla contiene los valores del coeficiente de Silhouette, la precisión, la exhaustividad y el valor-F BCubed de cada algoritmo aplicado sobre todos los datasets. De esta forma pudimos decidir rápidamente sobre qué datasets se comportaba mejor cada uno de los algoritmos.

K-Means

| Algoritmo | Silhouette | Valor-F | Precisión | Exhaustividad |
|---|---|---|---|---|
| K-Means | 0.108 | 0.61 | 0.492 | 0.802 |
| K-Means Acortado | 0.082 | 0.61 | 0.491 | 0.804 |
| K-Means Sin país, sexo ni edad | 0.14 | 0.609 | 0.491 | 0.8 |
| K-Means Sin país, sexo ni edad acortado | 0.116 | 0.602 | 0.485 | 0.793 |
| K-Means Sin sintomas ni signos | 0.122 | 0.612 | 0.496 | 0.801 |
| K-Means Sin sintomas ni signos acortado | 0.097 | 0.609 | 0.491 | 0.801 |
| K-Means Sin país, sexo, edad, sintomas ni signos | 0.169 | 0.605 | 0.49 | 0.793 |
| K-Means Sin país, sexo, edad, sintomas ni signos acortado | 0.187 | 0.247 | 0.52 | 0.162 |
| K-Means Elementos diagnósticos principales | 0.132 | 0.61 | 0.494 | 0.799 |
| K-Means Selección atributos CFS | 0.358 | 0.289 | 0.582 | 0.192 |
| K-Means Selección atributos FCBF | 0.693 | 0.337 | 0.532 | 0.246 |

Figura B.5: Tabla resumen por algoritmo