Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA DE SEGOVIA

Grado en Ingeniería Informática de Servicios y Aplicaciones

Construcción de un sistema de asistencia al operador de aprendizaje semi-autónomo

---

Alumno: Adrián Manzano Santos

Tutores: Anibal Bregón Bregón y Álvaro García García

Construcción de un sistema de asistencia al operador de aprendizaje semi-autónomo

Adrián Manzano Santos

Agradecimientos

A mi familia, en especial a mis padres, por su apoyo mostrado estos años. A la Fundación Cidaut y a mis compañeros, por proponerme y ayudarme con este proyecto. Y, especialmente, a Aníbal y Álvaro, por proponer y hacer posible este trabajo y por toda su ayuda prestada.

Resumen

En el contexto de la Industria 4.0, y en plena revolución digital, se desea la extracción de información del mundo real y su traslado a un mundo virtual. En este mundo virtual, la información es procesada para extraer conclusiones que se traducen en decisiones en el mundo real.

En este contexto, se presenta el estudio y desarrollo de una aplicación capaz de procesar la información recogida por sensores conectados, estructurarla mediante series temporales y clasificarla para la detección de anomalías. El objeto de estudio es una máquina fresadora CNC de más de veinte años de servicio, en pleno funcionamiento y sensorizada mediante una instrumentalización no intrusiva.

Para ello, se presenta un estudio sobre series temporales y se adapta el algoritmo de *clustering* de k-medias (*k-means*) a su aplicación sobre series temporales. Para aplicar el algoritmo de k-medias sobre un conjunto de datos, son necesarios dos componentes: una medida de comparación y una función promedio.

Como medida de comparación se estudia la "distancia" DTW (*Dynamic Time Warping*), que permite comparar series de distinta longitud (duración) y detectar eventos con distinto momento de ocurrencia temporal y/o distinta duración que se corresponden a una misma acción. Como función promedio se analiza la media de Fréchet, hallada mediante el algoritmo DBA (*Dynamic Barycenter Average*), y que a su vez se apoya en la DTW.

Adaptado el algoritmo de k-medias a su aplicación sobre series temporales, se propone un modelo de clasificación híbrido de dos etapas: en primer lugar divide el conjunto de las series atendiendo a atributos estáticos (duración) por el algoritmo de k-medias tradicional, y para cada subconjunto obtenido clasifica por el algoritmo de k-medias aplicado sobre las series temporales bajo la DTW.

Finalmente, se presenta una implementación escalable de aplicación de la teoría desarrollada anteriormente sobre la máquina fresadora: Sistema de Detección de Anomalías. El trabajo concluye con su completo desarrollo e integración dentro del proyecto IDIGIT4L de la Fundación CIDAUT.

Palabras clave: serie temporal, *Dynamic Time Warping* (DTW), clasificación no supervisada (*clustering*), algoritmo, algoritmo de k-medias, algoritmo de medias móviles, algoritmo DBA.

Abstract

In the context of Industry 4.0, and in the midst of the digital revolution, you want the extraction of information from the real world and its transfer to a virtual world. In this virtual world, information is processed to draw conclusions that translate into decisions in the real world.

In this context , this report presents the study and development of an application capable of processing the information collected by certain sensors, structuring it through time series and classifying it for the detection of anomalies. The object of this study is a CNC milling machine with more than twenty years of service, fully operational and sensorized by means of non-intrusive instrumentation.

To do this, a study on time series is presented and adapted the k-means *clustering* algorithm, to be used over time series. To apply the k-means algorithm to be used over a data set, two components are necessary: a comparison measure and an average function.

As a measure of comparison, the "distance" DTW (*Dynamic Time Warping*) is studied, which allows comparing series of different length (duration) and detecting events with different times of occurrence and/or different duration that correspond to the same action. As an average function, we analyze the Fréchet mean, found using the DBA algorithm (*Dynamic Barycenter Average*), which is supported by the DTW.

Once the k-means algorithm has been adapted to its application over time series, a two-stage hybrid classification model is proposed: first, it divides the set of series according to static attributes (duration) by the k-means algorithm. traditional, and for each subset obtained it classifies by the algorithm of k-means applied to the time series under the DTW.

Finally, a scalable implementation of the theory developed above on the milling machine is presented: Anomaly Detection System. The work concludes with its complete development and integration within the CIDAUT Foundation project IDIGIT4L.

Key words: time series, *Dynamic Time Warping* (DTW), clustering, algorithm, k-means, moving average algorithm, DBA algorithm.

Índice general

1. Introducción y objetivos **1**
1.1. Motivación . . . 2
1.2. Objetivos . . . 3
1.3. Organización de la memoria . . . 3

2. Metodología de trabajo y estado actual **6**
2.1. Entregables del proyecto . . . 6
2.2. Planificación temporal del proyecto . . . 7
2.3. Herramientas . . . 8
2.4. Presupuesto económico . . . 9

3. El objeto de estudio: la máquina fresadora **11**
3.1. El trabajo diario en la máquina fresadora . . . 11
3.2. La instrumentación de sensores . . . 13
3.3. La base de datos . . . 15
3.4. Resultados relevantes en actividades previas del proyecto iDIGIT4L . . . 18

4. Fundamento teórico **20**
4.1. Fundamentos sobre series temporales . . . 20
4.1.1. Componentes de una serie temporal: el modelo aditivo . . . 21
4.1.2. Series temporales estacionarias . . . 21
4.1.3. Determinar la tendencia de una serie: el algoritmo de medias móviles . . . 22
4.2. El algoritmo de k-medias . . . 23
4.2.1. Introducción . . . 23
4.2.2. Algoritmo de k-medias versión tradicional . . . 24
4.3. Medida de comparación de series temporales: La DTW . . . 25
4.3.1. La problemática en la comparación de series temporales. Deformaciones en el tiempo . . . 26
4.3.2. La distancia DTW . . . 27
4.3.3. Algoritmo para el cálculo de la distancia DTW: la matriz DTW . . . 28
4.3.4. La curva de deformación y las asociaciones entre índices . . . 29
4.4. Función promedio. Media de Fréchet DBA . . . 30
4.5. Algoritmo de k-medias aplicado a series temporales . . . 31

4.6. Los diferentes enfoques a la clasificación de series temporales 32

5. Estudio del problema y definición de un Modelo 34
5.1. Análisis de la información 34
5.2. Hipótesis de la estacionalidad sobre las series temporales 38
5.3. Arquitectura del Modelo 40
5.3.1. Los parámetros del modelo 40
5.3.2. El proceso de evaluación según un modelo 41

6. Arquitectura del proyecto 43
6.1. Una aplicación de dos sistemas 43
6.2. Características principales 44
6.3. Módulos de cada sistema 44

7. Análisis y diseño 49
7.1. Requisitos de Información 49
7.1.1. La base de datos completa 51
7.1.2. Descripción de los requisitos de información 53
7.2. Funcionalidades del sistema: Requisitos de Usuario 57
7.2.1. Actores del sistema 57
7.2.2. Requisitos de Usuario 58
7.2.3. Casos de uso 59
7.3. Diseño 73
7.3.1. Representación lógica de la información 73
7.3.2. Arquitectura lógica del modelo 74
7.4. Reglas de Negocio 79
7.5. Atributos de calidad 79

8. Desarrollo del proyecto 80
8.1. Etapa cero 81
8.1.1. Requisitos Funcionales 82
8.1.2. Detalles de la implementación 82
8.2. Etapa primera 82
8.2.1. Requisitos Funcionales 83
8.2.2. Detalles de la implementación 83
8.2.3. Validación y pruebas 85
8.3. Etapa segunda 85
8.3.1. Requisitos Funcionales 86
8.3.2. Detalles de la implementación 87
8.3.3. Diseño de las interfaces 87
8.3.4. Validación y pruebas 87
8.4. Tercera etapa 88
8.4.1. Requisitos Funcionales 89
8.4.2. Detalles de la implementación 89

8.4.3. Diseño de las interfaces . . . 91
8.4.4. Validación y pruebas . . . 91

9. Estudio experimental 94
9.1. La configuración para la clasificación . . . 94
9.2. Resultados de la clasificación . . . 97
9.2.1. El valor de tolerancia . . . 97
9.3. Evaluación del modelo . . . 99
9.4. Ventajas del modelo híbrido . . . 100

10. Conclusiones y líneas de trabajo futuras 102
10.1. Conclusiones y consecución de objetivos . . . 102
10.2. Aprendizaje personal . . . 103
10.3. Líneas de trabajo futuras . . . 104

A. Contenido entregado y manual de uso 106
A.1. Contenido entregado . . . 106
A.2. Manuales . . . 108
A.2.1. Requisitos para poder utilizar el sistema . . . 108
A.2.2. Manual de ejecución del Sistema de Detección de Anomalías . . . 108
A.2.3. Manual de uso de la aplicación . . . 109

B. Implementación de los algoritmos en pseudocódigo 115
B.1. Algoritmo de Medias Móviles . . . 115
B.2. Implementación de la DTW versión primera . . . 116
B.3. Implementación del algoritmo para hallar la curva de deformación . . . 117
B.4. Implementación del algoritmo para determinar las asociaciones de índices . . . 118
B.5. Implementación algoritmo DBA . . . 119
B.6. Implementación del algoritmo de clustering sobre vectores . . . 120
B.7. Implementación del algoritmo de clustering sobre series temporales . . . 123

C. Fundamento sobre series temporales 126
C.1. El concepto de serie temporal . . . 126
C.1.1. Introducción . . . 126
C.1.2. Componentes de una serie temporal: El modelo aditivo . . . 127
C.1.3. Series temporales con modelo estacionario . . . 130
C.1.4. Técnicas para determinar la tendencia. El algoritmo de medias móviles . . . 132
C.2. La distancia DTW . . . 134
C.2.1. Caminos de deformación y su coste . . . 134
C.2.2. La distancia DTW como coste mínimo . . . 136
C.2.3. Comportamiento de la distancia DTW frente a deformaciones . . . 137
C.2.4. Restricciones que debe verificar la matriz DTW . . . 139
C.2.5. Teorema de la curva de deformación . . . 140
C.2.6. Algoritmo para determinar las asociaciones de índices . . . 142

C.2.7. Ejemplo DTW 142
C.2.8. Otras versiones del Algoritmo DTW 144

D. Fundamento sobre *clustering* **150**
D.1. Concepto de *clustering* 150
D.2. Algoritmos de clasificación (*clustering*) 150
D.3. Algoritmo de k-medias versión de la matriz de pertenencia 151
D.4. Media de Fréchet y algoritmo DBA 153
D.4.1. Construcción del algoritmo de k-medias aplicable a series temporales . . . 155
D.5. Problemas del algoritmo de k-medias 157

E. Ejemplo de realizaciones de ciertas actividades **160**

F. Resultados del estudio experimental **166**

Índice de figuras

2.1. Diagrama de Gantt donde se recoge la planificaicón temporal 8

3.1. Equipo con pantalla táctil con el sistema de control instalado 16
3.2. Diagrama entidad-relación de la base de datos, representando una imagen de esta antes de comenzar el proyecto 17
3.3. Representación de la corriente medida por el sensor de corriente de fase 1 durante el arranque de la máquina fresadora identificando sus fases anteriormente descritas. 18

4.1. Ejemplo del efecto de deformación en el tiempo 26

5.1. Representación de una operación de contorneado sobre el acero 35
5.2. Ocurrencias de cada tipo de operación con cada material 36
5.3. Histograma de las duraciones de las operaciones sobre el acero con duración inferior a una hora y 30 minutos (izquierda) y diagrama de barras con las duraciones agrupadas (derecha) 36
5.4. Diagrama de barras del número de operaciones resultantes: en azul sin imponer ningún tipo de restricción, en naranja tras exigir que la duración sea mayor de 5 minutos y menor de una hora y en verde tras exigir también que sean posteriores al 1 de o de 2019 37
5.5. Agrupaciones de los arranques según su duración (izquierda) y diagrama de barras del número de operaciones resultantes (derecha): en azul sin imponer ningún tipo de restricción, en naranja tras exigir que la duración sea mayor de 3 minutos y menor de una hora y en verde tras exigir también que sean posteriores al 1 de septiembre de 2019 38
5.6. En gris la serie temporal original y en rojo la serie con la estacionalidad suprimida por el algoritmo de medias móviles con tamaño de ventana 10 en la versión centrada. 38
5.7. Representación del resultado de la prueba de estacionalidad. 39
5.8. Esquema del modelo de clasificación híbrido planteado 41

6.1. Diagrama de estructura del Sistema de Detección de Anomalías 46
6.2. Diagrama de secuencia para la característica CAR-01 47
6.3. Diagrama de secuencia para la característica CAR-02 47
6.4. Diagrama de secuencia para la característica CAR-03 48

7.1. Diagrama Entidad-Relación para modelar los requisitos de información y sus relaciones . . . 51
7.2. Diagrama entidad-relación correspondiente a la base de datos completa . . . 52
7.3. Diagrama de estados para el requisito de información **RI-05** . . . 56
7.4. Diagrama de casos de uso del Sistema de Detección de Anomalías . . . 60
7.5. Diagrama de actividad correspondiente al caso de uso **RU-01** . . . 62
7.6. Diagrama de Actividad para el caso de uso **RU-02** . . . 64
7.7. Diagrama de actividad para el caso de uso **RU-03** . . . 66
7.8. Diagrama de actividad para el caso de uso **RU-04** . . . 68
7.9. Diagrama de actividad para el caso de uso **RU-05** . . . 70
7.10. Diagrama de actividad para el caso de uso **RU-06** . . . 72
7.11. Diagrama de clases para la estructuración de las actividades. . . . 75
7.12. Diagrama de clases para el diseño del modelo de clasificación . . . 78
7.13. Diagrama de clases completo del sistema de detección de anomalías . . . 78

8.1. Diagrama de navegación para las interfaces de la etapa segunda. . . . 88
8.2. Diagrama de presentación para las interfaces desarrolladas en la segunda etapa . 88
8.3. Diagrama de navegación de interfaces completo . . . 91
8.4. Diagrama de presentación de interfaces completo . . . 92

9.1. Ejemplo de aplicación del método del codo. La solución es de 5 clústeres. . . . 95
9.2. Representación de la distancia intra-clúster para diferentes configuraciones de número de clústeres para las duraciones, para el tipo de operación de contorneado sobre acero. . . . 95
9.3. Representación de la distancia intra-clúster como función objetivo, resultado del estudio de la configuración para la clasificación . . . 96
9.4. Representación del porcentaje de actividades asignados a cada clúster final del modelo para cada *opermat*. . . . 98
9.5. Representación del porcentaje de actividades consideradas normales y anómalas para diferentes valores de tolerancia discriminando por el *opermat* . . . 98
9.6. Representación de los resultados de evaluar las actividades . . . 99
9.7. Representación de los tiempos requeridos en la construcción de modelos con diferentes configuraciones de la operación de contorneado sobre el acero . . . 101

A.1. Documentos incluidos en `SistemaDeteccionAnomalias.rar` . . . 107
A.2. Opciones ofrecidas por la aplicación . . . 109
A.3. Página de Bienvenida del SDA . . . 110
A.4. Página de "Funcionamiento" del SDA . . . 110
A.5. Página de "Modelo" del SDA . . . 111
A.6. Página de "Modelo" del SDA con alguna entrada errónea . . . 111
A.7. Página de "Estudio experimental" con errores en los campos de entrada . . . 112
A.8. Página de "Estudio experimental" con problemas en la conexión con la BBDD . . . 112
A.9. Página de "Estudio experimental" . . . 113
A.10. Interfaz de resultado del estudio experimental en el SDA . . . 114

A.11.Página "estudio tolerancia" del SDA 114

C.1. Ejemplo de la tendencia de una serie temporal 128
C.2. Serie temporal definida como $x_k = 2,75k + 10sen(k) + 10 \quad k = 1, 2, \ldots, 51$ 129
C.3. Ejemplo de serie temporal con modelo estacionario de ruido blanco 131
C.4. Ejemplo de series temporales con modelo no estacionario de camino aleatorio . . 131
C.5. Ejemplo de serie temporal con tendencia determinista (izquierda) y con tendencia evolutiva (derecha) 133
C.6. Comparación de la asociación cronológica (A) con una asociación con deformación en el tiempo (B). Las líneas grises punteadas indican las asociaciones. 137

D.1. Ejemplo de *clustering* con la flor de iris: representación de los datos (izquierda) y resultado del algoritmo (derecha) 153
D.2. Representación de las series temporales a promediar *(a)* y del promedio obtenido mediante el algoritmo DBA *(b)*. 154
D.3. Representación de las series temporales **X**, **Y** y **Z** con sus asociaciones de índices respecto de **X** (izquierda) y junto a su media **C** (derecha) 155
D.4. Ejemplo de agrupaciones complejas, cuya solución falla por k-medias. 158
D.5. Ejemplo de clustering incorrectos con la flor de iris: agrupación con dos clústeres (izquierda) y agrupación con los clústeres iniciales erróneos (derecha) 158

E.1. Representación de las mediciones realizadas por el sensor de corriente de fase 1 para la operación tipo planeado sobre el acero para algunas realizaciones concretas y resultado de clasificar dichas realizaciones por el modelo correspondiente 161
E.2. Representación de las mediciones realizadas por el sensor de corriente de fase 1 para la operación tipo contorneado sobre el acero para algunas realizaciones concretas y resultado de clasificar dichas realizaciones por el modelo correspondiente 162
E.3. Representación de las mediciones realizadas por el sensor de corriente de fase 1 para la operación tipo fresado sobre el acero para algunas realizaciones concretas y resultado de clasificar dichas realizaciones por el modelo correspondiente 163
E.4. Representación de las mediciones realizadas por el sensor de corriente de fase 1 para la operación tipo taladrado sobre el acero para algunas realizaciones concretas y resultado de clasificar dichas realizaciones por el modelo correspondiente 164
E.5. Representación de las mediciones realizadas por el sensor de corriente de fase 1 para lel arranque de la máquina fresadora y resultado de clasificar dichas realizaciones por el modelo correspondiente 165

F.1. Representación de la distancia intra-clúster bajo la DTW como función objetivo para la operación de taladrado sobre acero 166
F.2. Representación de la distancia intra-clúster bajo la DTW como función objetivo para la operación de fresado sobre acero. 167
F.3. Representación de la distancia intra-clúster bajo la DTW como función objetivo para la operación de planeado sobre acero 168
F.4. Representación de la distancia intra-clúster bajo la DTW como función objetivo para para la operación de contorneado sobre acero 169

Índice de tablas

1.1. Objetivos del proyecto 4

2.1. Presupuesto del proyecto 10

3.1. Operaciones realizables en la fresadora y sus códigos 12
3.2. Materiales de uso en la fresadora y sus códigos 12
3.3. Herramientas de la fresadora y sus códigos 13
3.4. Relación de las operaciones, herramientas y materiales 14
3.5. Unidades de medida y precisión de los sensores 14
3.6. Campos e información en la tabla `CNC_Funcionamiento` 15
3.7. Campos e información de la tabla *tiempoOperaciones* 17

6.1. Características del Sistema de Detección de Anomalías 44

7.1. Requisitos de Información del Sistema de Detección de Anomalías 50
7.2. Relaciones entre los requisitos de información 50
7.3. Campos e información de la tabla `CNC_Incidencias` 53
7.4. Parámetros de configuración **RI-04** 54
7.5. Parámetros de configuración **RI-05** 55
7.6. Parámetros de configuración **RI-06** 56
7.7. Parámetros de configuración **RI-06** 57
7.8. Actores del Sistema de Detección de Anomalías 58
7.9. Actores del Sistema de Detección de Anomalías 59
7.10. Descripción del requisito de usuario **RU-01** 61
7.11. Descripción del requisito de usuario **RU-02** 63
7.12. Descripción del requisito de usuario **RU-03** 65
7.13. Descripción del requisito de usuario **RU-04** 67
7.14. Descripción del requisito de usuario **RU-05** 69
7.15. Descripción del requisito de usuario **RU-06** 71
7.16. Reglas del negocio 79
7.17. Atributos de calidad 79

8.1. Requisitos Funcionales de la Etapa 0 82
8.2. Requisitos Funcionales de la Etapa 1 84

8.3. Prueba de caja negra sobre la generación de los modelos de clasificación 85
8.4. Prueba de caja negra sobre la evaluación de las actividades 86
8.5. Requisitos Funcionales de la Etapa 0 86
8.6. Prueba de caja negra sobre la generación de los modelos de clasificación con configuración propia. 89
8.7. Prueba de caja negra sobre la evaluación de las actividades 90
8.8. Requisitos Funcionales de la Etapa 0 90
8.9. Prueba de caja negra sobre el estudio para la clasificación 92
8.10. Prueba de caja negra sobre el estudio del valor de tolerancia I 93
8.11. Prueba de caja negra sobre el estudio del valor de tolerancia II 93

E.1. Resultados de evaluar las actividades seleccionadas como ejemplo por su correspondiente modelo según los identificadores dados en las *Figuras E.1, E.2, E.3, E.4 y E.5* 160

Capítulo 1

Introducción y objetivos

Actualmente estamos inmersos en la *Cuarta Revolución Industrial*, también conocida como *Industria 4.0* [18], que está considerada la etapa más importante desde la revolución industrial del siglo XVIII. Esta cuarta etapa está cambiando la forma en la que los negocios operan y los entornos en los que compiten. El concepto de Cuarta Revolución Industrial es bastante actual y fue acuñado por Klauss Schwab [19] en el Foro Económico Mundial de 2016, siendo las primeras referencias a dicha idea posteriores al año 2010.

Mientras que la Tercera Revolución Industrial está caracterizada por la *era digital*, la Industria 4.0 busca combinar técnicas avanzadas de producción con las nuevas tecnologías tales como la *robótica*, la *Inteligencia Artificial* o el *Internet de las Cosas* entre otros, recogiendo la tendencia actual a la automatización y el intercambio de información. Para mejorar los procedimientos industriales, se comienza con una captura de la información del entorno físico traduciéndola a un entorno digital. En dicho entorno digital, la información es interpretada de forma analítica y analizada objetivamente para extraer unos resultados o conclusiones que finalmente se traducen en decisiones en el mundo real.

El presente Trabajo de Fin de Grado se engloba dentro del proyecto **IDIGIT4L**[1] de la *Fundación Cidaut* cuyo objetivo principal es la investigación de un ecosistema de digitalización en el cual intervienen personas, objetos y sistemas con el fin de transformar procesos industriales de forma inteligente y predictiva. Gracias a la convergencia del mundo físico y del mundo digital, se presenta el concepto de *Gemelo Digital distribuido*, basado en la virtualización de activos y el conocimiento adaptativo de las operaciones que permita mejorar los procesos y la calidad de los productos.

[1] "I+D para la transformación de los sectores industriales e infraestructuras: Digitalización y su implementación inteligente y predictiva iDIGIT4L", N.º de Exp.: CCTT1/17/VA/0007. Proyecto financiado por el Instituto para la Competitividad Empresarial de Castilla y León (Junta de Castilla y León), cofinanciado con Fondos FEDER, a través de la línea: "PROYECTOS I+D APLICADOS REALIZADOS POR CCTT. Convocatoria 2017".

Con este objetivo, se han implementado soluciones [16] y [17] apoyadas en la retroactualización y convergencia de activos industriales, independientemente de su nivel de digitalización, con el uso de sensores no intrusivos. En particular, se ha trabajado con una **máquina fresadora CNC** (control numérico por computadora) modelo Nicolás Correa CF20 a la cual se conectaron sensores no intrusivos para obtener valores de vibración, temperatura y consumo energético. Estos sensores permiten caracterizar el estado de la máquina fresadora con el objetivo de mejorar el rendimiento y la calidad de los productos obtenidos. Se pretende estudiar un entorno de fabricación industrial, introduciendo mejoras basadas en la digitalización. Para ello se ha desarrollado un software predictivo capaz de caracterizar el funcionamiento correcto del sistema y detectar desviaciones y/o anomalías durante el funcionamiento de la máquina fresadora CNC.

Por tanto, este trabajo, lejos de ser un desarrollo e implementación tradicional de una aplicación, recoge un estudio completo del problema y de la solución adoptada así como el diseño e implementación de una aplicación capaz de analizar y procesar la información extraída por los sensores y valorar objetivamente la bonanza de las acciones realizadas, que se conoce como sistema de detección de anomalías.

El modelado de la información se realiza mediante *series temporales*, es decir, conjunto de observaciones etiquetadas en el tiempo y para el análisis se recurren a técnicas (algoritmos) de aprendizaje automático no supervisado por clasificación de conjuntos no etiquetados o *clustering* con la finalidad de encontrar los "patrones de funcionamiento usuales" y detectar aquellos "patrones raros" o de poca ocurrencia. En una segunda retroalimentación, el operario de la máquina fresadora trata de identificar y etiquetar la causa de las ocurrencias y patrones poco usuales para que, en un futuro, cuando se vuelva a producir alguno, el sistema pueda inferir la causa y/o recomendar las acciones a llevar a cabo.

Finalmente, dentro del proyecto IDIGIT4L, se implementan soluciones de interacción hombre-máquina apoyadas en la Realidad Aumentada para ser capaces de guiar al operario en la realización de las acciones que recomiende el sistema de detección de anomalías.

1.1. Motivación

La motivación del presente trabajo es el contexto actual de evolución industrial, que obliga a adaptarse a las nuevas condiciones del mercado [21] y para lo cual se empieza a trabajar en marzo del 2018 en el proyecto IDIGIT4L [2] en el área de *TIC e Industria 4.0* de la Fundación Cidaut, y cuyo concepto se remonta al año 2017.

[2]Fecha de inicio 05/03/2018

En este proyecto se pueden diferenciar tres conceptos clave: medición de la información de forma no intrusiva; virtualización con análisis y evaluación de la información; y notificación de incidencias y ayuda al usuario, con las acciones a realizar para mejorar el rendimiento o evitar paradas no planificadas por incidencias en los equipos de producción.

Motivada por cubrir la parte de virtualización y análisis de la información extraída se desarrolla esta memoria, en la que se recoge un estudio experimental del problema de clasificación no supervisada y un desarrollo e implementación escalable de un sistema que satisfaga los requerimientos de valoración y procesamiento recogidos en los objetivos del presente trabajo (véase *Sección 1.2*). Dicha valoración se basa, como se mencionó previamente, en la clasificación no etiquetada de las acciones realizadas con objeto de identificar aquellas que tienen una mayor ocurrencia (consideradas "normales") y aquellas de una baja ocurrencia o nula reiteración (consideradas "ocasionales") y darlas un significado o sentido a su aparición basado en la experiencia del operario encargado de la máquina fresadora CNC.

1.2. Objetivos

El presente trabajo de fin de grado estudia el ámbito de la Industria 4.0 aplicado al mantenimiento predictivo de una máquina CNC sobre la que se han instrumentado mediciones no intrusivas. Este trabajo se enmarca dentro del alcance de un proyecto de I+D, en el que se buscan realizar mediciones de un sistema industrial, clasificar la información recibida e interactuar con el operador indicándole los posibles errores y sus soluciones mediante un monitor y unas instrucciones dadas por un sistema de realidad aumentada.

Por tanto, se pretende desarrollar un gran módulo capaz de procesar la información medida en un sistema de fabricación industrial y determinar tanto su grado de bonanza como las posibles mejoras que el operador pueda realizar. Para ello, se identifican lo siguientes objetivos del proyecto recogidos en la *Tabla 1.1*.

1.3. Organización de la memoria

En esta sección se ilustra como se ha estructurado el presente documento. Este trabajo ha sido dividido en diferentes capítulos que tratan los distintos aspectos del proyecto.

En primer lugar en el *Capítulo 2*, se identifican los entregables del proyecto (*Sección 2.1*), se realiza una planificación temporal inicial (*Sección 2.2*), se identifican las herramientas utilizadas (*Sección 2.3*) y se introduce el caso de estudio (*Capítulo 3*) describiendo en detalle la máquina fresadora CNC.

Id objetivo	Objetivo
Obj-01	Estudiar y analizar el funcionamiento de la máquina fresadora para comprender el problema e interpretar la información obtenida
Obj-02	Análisis y clasificación del histórico de datos disponible de la máquina fresadora para la generación de un modelo de clasificación
Obj-02.1	Desarrollar e implementar algoritmos para el tratamiento y procesado de la información en bruto leída por los sensores
Obj-02.2	Estudiar la información procesada para elegir los datos asociados de utilidad.
Obj-02.3	Estudiar de la información útil para elegir una técnica de clasificación adecuada al problema y a los datos
Obj-02.4	Implementar la técnica de clasificación elegida
Obj-02.5	Determinar la mejor configuración para los parámetros de la técnica de clasificación y generar el modelo de clasificación
Obj-02.6	Estudiar el modelo de clasificación generado, su validez y los resultados obtenidos.
Obj-03	Desarrollar una implementación escalable del algoritmo elegido, capaz de realizar análisis predictivo sobre los datos en vivo aplicando los modelos obtenidos
Obj-03.1	Desarrollo de un sistema capaz de conectar con la base de datos, leer la información y cargar las incidencias detectadas.
Obj-03.2	Desarrollo de un script capaz de general los modelos a partir de la información seleccionada y realizar su mantenimiento cuando proceda.
Obj-03.3	Desarrollo de un algoritmo capaz de aplicar los modelos a la información preprocesada y valorar de forma autónoma el resultado.

Tabla 1.1: Objetivos del proyecto

En el *Capítulo 4* y en los *Apéndices C y D* se presentan los conceptos que fundamentan y permiten la construcción de la aplicación. Así, en las *Secciones 4.1 y 4.3* se introduce el concepto de serie temporal y la medida de comparación *Dynamic Time Warpping* (DTW) utilizada sobre dichas series, junto a otras propiedades de enorme interés. Por otro lado, en las *Secciones 4.2 y 4.5* se describe el algoritmo de k-medias (*k-means*), y se presentan todas las modificaciones y técnicas necesarias para su aplicación sobre series temporales.

Capítulo 5 presenta un análisis sobre la información recogida y concluye con la identificación y determinación de la arquitectura de un Modelo de Clasificación, derivado de la técnica de k-medias. Este modelo se caracteriza por determinados parámetros, que son estudiados en el *Capítulo 9* con el objetivo de elegir la mejor configuración de modelo.

Finalmente, todo el estudio del problema concluye en la realización de una aplicación, denominado Sistema de Detección de Anomalías (SDA). Todo su desarrollo se trata en los *Capítulos 6, 7 y 8.*

En el *Capítulo 6* se describe la arquitectura del proyecto. En el *Capítulo 7* se recogen los requisitos que deben satisfacerse, las funcionalidades que se esperan y se realiza el diseño de la arquitectura lógica del modelo y de las estructuras para regularizar la información recogida por los sensores. Y, en el *Capítulo 8* se recoge el desarrollo evolutivo-incremental del Sistema de Detección de Anomalías.

Capítulo 2

Metodología de trabajo y estado actual

Una de las principales tareas de cualquier proyecto es el estudio y elección de las técnicas y herramientas que serán necesarias para el correcto desarrollo de este, y que permitan cumplir los objetivos definidos (véase *Tabla 1.1*). En este capítulo se recoge toda esta información de interés.

En este capítulo se exponen tanto la metodología planteada para desarrollar el proyecto como la planificación temporal y las herramientas necesarias. En particular, dado que este proyecto es parte de otro mayor englobado dentro de las *prácticas en la Fundación Cidaut* se requiere tener completado ciertas partes del mismo en momentos concretos.

También es importante destacar que el trabajo parte del estudio del conocimiento generado a partir de las conclusiones de un proyecto anterior realizado sobre la máquina fresadora CNC así como la estructura de los componentes ajenos al proyecto con los que se deben interactuar. A esto en particular se le dedicará el *Capítulo 3*.

2.1. Entregables del proyecto

1. Un **compendio bibliográfico** que comprende un estudio detallado del problema de clasificación de las acciones realizadas en la máquina fresadora CNC y de las técnicas matemáticas y algoritmos aplicados en su resolución. Comprende el *Capítulos 4* y los *Apéndices C y D* de la presente memoria, donde se detallan las técnicas para la representación abstracta de la información y un algoritmo de *clústering* para su posterior clasificación.

2. Un **estudio del problema** donde se analice la información recogida por los sensores y se proponga una arquitectura para un modelo de clasificación adecuado y optimizado (*objetivo primero* **Obj-01**). Comprende los *Capítulos 5 y 9*.

3. Una **aplicación desarrollada en *Python*** capaz de procesar la información para generar los modelos (*objetivo segundo* **Obj-02**) y valorar los resultados produciendo indicaciones al operario (*objetivo tercero* **Obj-03**). Se trata en los *Capítulos 6, 7 y 8* y compone lo que se denominará **Sistema principal**.

4. Una **aplicación desarrollada en *Python*** utilizada para la realización de un estudio experimental sobre los datos con el fin de determinar los parámetros óptimos que requieren predefinir el algoritmo de clasificación utilizado. Se trata en los *Capítulos 6, 7 y 8* y compone lo que se denominará **Sistema secundario**.

5. Un **informe de resultados** en forma de análisis comparativo en el que bajo la hipótesis "la proporción de actividades anómalas es constante en el tiempo" se valore la consistencia del modelo. Esto se recoge en las *Secciones 9.2, 9.3 y 9.4*.

2.2. Planificación temporal del proyecto

En lo que respecta al ciclo de vida del proyecto se identifican cinco fases.

1. **Familiarización con el proyecto general.** En esta primera fase se realiza un estudio general de la información recogida y de la máquina fresadora para conocer su funcionamiento y dar significado a la información. Se finaliza estudiando los resultados de un proyecto anterior.

2. **Análisis y elección de las técnicas: estudio preliminar de los algoritmos.** Una vez estudiada la información, se determina que se trabajará con series temporales y se realiza una breve búsqueda de aquellas técnicas de clasificación que mejor se adapten al problema. El resultado es el algoritmo de k-medias bajo la distancia *Dynamic Time Warping* (DTW) (consúltense [5] y [6]). Por último se estudiará el comportamiento de dicho algoritmo en función de la cantidad de información utilizada, para lo cual se recurre al módulo de *Python* `tslearn` (consúltese [1]).

3. **Implementación y estudio del modelo de la máquina fresadora CNC.** Aplicando el algoritmo de k-medias bajo la distancia DTW se pretende construir un modelo de clasificación y comportamiento. Para ello se valorará que parte de la información recopilada es realmente información útil, y va a ser utilizada para la clasificación, y se analizan sus repercusiones en función de su dominio de valores. Finalmente, se extraen varios patrones que representan el conjunto de las actividades estudiadas. Se crean varios modelos diferenciando según dos atributos concretos, que son la operación realizada en la máquina y el material con el que se trabaja. Estos atributos se detalla en el *Capítulo 3*, y el modelo de clasificación elegido se recoge en el *Capítulo 5*.

4. **Implementación de una aplicación capaz de evaluar el funcionamiento de la fresadora CNC.** Con los modelos ya creados, se desarrolla e implementa una aplicación en *Python* capaz de leer la información de los sensores en vivo y clasificar y evaluar según el modelo correspondiente dado en el punto tercero, determinado por la que operación se está realizando y sobre que material se está trabajando.

5. **Validación de los resultados de las evaluaciones realizadas.** Se contrasta las evaluaciones obtenidas por la aplicación (punto cuarto) con la opinión fundada del operario. Con ello se buscan identificar las deficiencias de los modelos creados en el punto tercero así como recolectar nueva información que permitan en un futuro aconsejar al operario en caso de fallo y/o advertir cuando se produzcan desviaciones de lo que se considere un comportamiento estándar de la máquina fresadora CNC.

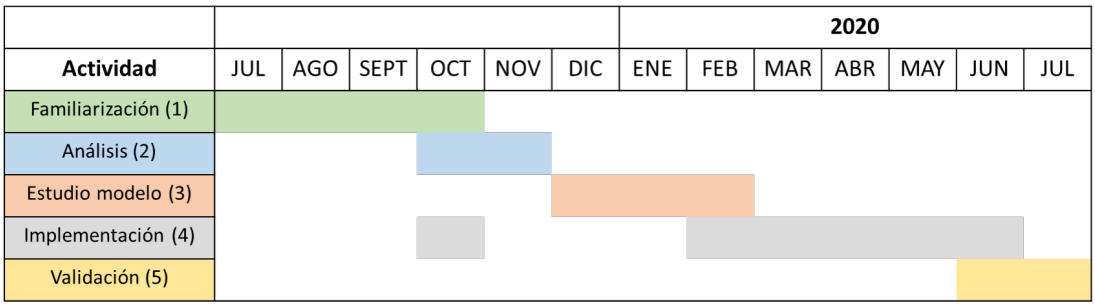


Figura 2.1: Diagrama de Gantt donde se recoge la planificaicón temporal

En cuanto a la planificación temporal, la fase de familiarización se desarrolla en los meses de julio a octubre. De noviembre a enero se realizan las fases de análisis de técnicas e implementación de modelos de comportamiento, concluyendo en el mes de febrero con los modelos y los patrones de las actividades realizadas en la fresadora CNC obtenidos a partir de los modelos de comportamiento. Por último, la fase de implementación de la aplicación de evaluación se extiende de marzo a junio solapándose con la fase de evaluación que comprende los meses de junio y julio.

2.3. Herramientas

Un requisito fundamental fijado por la Fundación Cidaut, es la implementación de la aplicación en *Python*. Además, la aplicación debe ser capaz de comunicarse adecuadamente con los módulos existentes en la infraestructura de aplicaciones y servicios del sistema de monitorización y recogida de datos para investigación. Las herramientas y *frameworks* utilizados son:

1. ***Jupyter Notebook*** y ***Spyder*** como entornos de trabajo. Como se pretende desarrollar una aplicación en *Python*, *Spyder* es adecuado como entrono de desarrollo mientras que *Jupyter Notebook* es adecuado para la representación y estudio de la información de funcionamiento y de los modelos producidos. Ambos entornos presentan una sencilla gestión de los módulos de *Python*, así como una interfaz amigable y una serie de herramientas que facilitan el desarrollo.

2. Los módulos *Python* **`numpy`** y **`pandas`** adecuados para estructurar y formatear la información extraída directamente de la base de datos y los módulos **`matlplotlib`** y **`plotly`** apropiado para la representación visual de la información.

3. Para la conexión con la base de datos y la realización de consultas se utiliza los módulos **`pyodbc`** y **`sqlalchemy`**, que establece de manera transparente al usuario la conexión con la base de datos tipo SQL.

4. El módulo *Python* **`sklearn`** que contiene los métodos de *machine learning* estándar, entre ellos el algoritmo de k-medias aplicado a conjuntos numéricos de igual dimensión y el módulo **`tslearn`** que implementa los principales métodos necesarios para trabajar con series temporales, entre ellos implementa la métrica *Dynamic Time Warping*, el promedio *DTW Barycenter Averaging* y el algoritmo de k-medias bajo dicha métrica.

5. El módulo *Python* **`Dash`** utilizado para el diseño e implementación de las interfaces e interactuar con el sistema como un servicio web.

6. El módulo *Python* **`statsmodels`** utilizado para el trabajo con series temporales (descomposición, test de estacionalidad, etc.).

2.4. Presupuesto económico

En este apartado se desglosan los costes y gastos del proyecto, tantos de recursos Software y Hardware como de humanos. El proyecto tiene fecha de inicio 1 de Julio de 2019 y fecha de fin 31 de Mayo de 2020. Incluyendo el proceso de documentación, la fecha de fin es 15 de Julio de 2020, considerando el mes de Junio como mes no laborable.

El presupuesto se recoge en la siguiente tabla (2.1).

Recurso	Coste total	Dedicación al proyecto	Coste real
Ordenador HP Omen	900€	10 %	90€
Ordenador Fundación CIDAUT	1000€	20 %	200€
Python y sus librerías	0€	100 %	0€
`Dash` MIT license	0€	100 %	0€
Jupyter Notebook	0€	100 %	0€
TeXStudio	0€	100 %	0€
Sistema de Sensores	1400€	100 %	1400€
Personal de desarrollo	1500€	100 %	1500€
TOTAL			3190€

Tabla 2.1: Presupuesto del proyecto

Capítulo 3

El objeto de estudio: la máquina fresadora

En este capítulo se describe el objeto de estudio y recoge cierta información de interés sobre su funcionamiento, la información disponible y resultados de interés del proyecto IDIGIT4L.

En la *Fundación CIDAUT* disponen de una máquina fresadora CNC (modelo Nicolás Correa CF20-18 de 1994) en pleno funcionamiento dedicada a la fabricación de piezas prediseñadas bajo demanda. Se ha elegido para el estudio debido a sus características: equipamiento de más de 20 años de servicio, no digitalizada y con órdenes de fabricación de diversa naturaleza donde cada pieza es única y diferente al resto de piezas fabricadas. Internamente, cada pieza es etiquetada por un identificador único denominado *orden de fabricación* y se elabora siguiendo unos **planos** determinados por el sistema de gestión de la oficina técnica.

3.1. El trabajo diario en la máquina fresadora

La máquina fresadora CNC, situada en el taller de fabricación que Fundación CIDAUT tiene en sus instalaciones del Parque Tecnológico de Boecillo permite realizar siete **operaciones** (véase *Tabla 3.1*) sobre cinco **materiales** (véase *Tabla 3.2*) diferentes. Una pieza se obtiene a partir de la realización de una o varias operaciones sobre un material concreto.

Cada material y operación se identifican por un código único que quedan recogidos en las *Tablas 3.2 y 3.1*, reservando el 0 para el estado ocioso de la máquina (sin material ni operación). Cada operación puede ser realizada con una o varias **herramientas**. Las herramientas quedan recogidas en la *Tabla 3.3*, y de igual manera, son identificadas mediante un código numérico.

Además, la herramienta que se usa para una operación concreta depende del tipo de material con el que se está trabajando. En la *Tabla 3.4* se recogen estas relaciones de operaciones y materiales con herramientas.

Operación	Código
Toma de ceros	1
Planeado	2
Contorneado o copiado	3
Mandrinado	4
Fresado	5
Taladrado	6
Especiales	7

Tabla 3.1: Operaciones realizables en la fresadora y sus códigos

Material	Código
Plástico	1
Aluminio	2
Acero	3
Acero Inox o 316	4
Otros	5

Tabla 3.2: Materiales de uso en la fresadora y sus códigos

El diseño y producción de piezas en la máquina fresadora es bajo demanda, es decir, cada pieza diseñada y producida lo es para cubrir una necesidad concreta, siendo poco probable que se vuelva a producir otra pieza igual en un corto periodo de tiempo. Por tanto, a la hora de comparar los datos de dos operaciones en la máquina hay que tener en cuenta que no son sobre el mismo diseño (plano) aunque se trate del mismo material. Por ejemplo, en la operación de taladrado sobre acero, un factor importante es el grosor del material bruto, que influye especialmente sobre la duración de la operación.

Herramienta	Código		Herramienta	Código
Centrador “bailarina”	1		⊘ 32 octogonal	2
⊘ 63 octogonal	3		⊘ 80 octogonal	4
⊘ 50 escuadral	5		⊘ 80 M4	6
⊘ 65 APKT invertido	7		⊘ 18	8
⊘ 20 APKT largo	9		⊘ 16 APKT	10
⊘ 18 APKT	11		⊘ 30 APKT	12
⊘ 16	13		⊘ 30	14
⊘ 32	15		⊘ 50	16
Mandrino	17		Cabeza mandrinadora	18
Portapinza ⊘ 0-3	19		Portapinza ⊘ 3-14	20
Cono ⊘ > 14	21		Achafado ⊘ 12	22
Achafado ⊘ 22	23		Turbo	24
Alineado-Reloj comparador	25		Portaherramienta Ni-Fresa	26
Portapinza-Fresa	27		⊘ 12	28
⊘ 18 largo	29		broca de plaguitas	30
⊘ 20	31			

Tabla 3.3: Herramientas de la fresadora y sus códigos

3.2. La instrumentación de sensores

Para la recolección de información sobre el funcionamiento de la máquina se dispone de una instrumentación de sensores capaces de realizar mediciones no intrusivas (es decir, el propio sensor no altera el funcionamiento de la componente ni su medición). Estos sensores miden el consumo, posición de la fresa, temperatura y sonido producido.

- **Acelerómetro**: determina la posición en la que se encuentra el mandrino respecto de un origen preestablecido sobre tres ejes con datos de cada coordenada en el espacio. De forma que el acelerómetro registra la posición en tres ejes denominadas respectivamente acelerómetro-eje-X, acelerómetro-eje-Y y acelerómetro-eje-Z.
- **Sonda de temperatura**: mide la temperatura en la zona de trabajo interior de la fresadora.
- **Sonda trifásica** : determina el consumo de la máquina fresadora. La máquina fresadora dispone de tres *relés* que realizan un consumo de corriente. Cada relé dispone de su propio sensor de corriente, denominados respectivamente sensor de fase de corriente 1, 2 y 3.
- **Detector de sonido**: mide los niveles de presión sonora (ruido) producido por la máquina fresadora en la zona interior de trabajo.
- **Sensor de la puerta**: determina si la puerta que delimita la zona de trabajo se encuentra abierta o cerrada.

Operación	Materiales	Herramientas
Toma de ceros	Indiferente	- Centrador "bailarina"
Planeado	Acero y Acero Inox	- ⊘ 32 ortogonal - ⊘ 63 ortogonal - ⊘ 80 ortogonal - ⊘ 50 escuadrar - ⊘ 80 M4 - ⊘ 65 APKT inverso
	Aluminio y Plásticos	- ⊘ 32 - ⊘ 50
Conotorneado/ Copiado	Acero y Acero Inox	- ⊘ 12 - ⊘ 18 APKT largo - ⊘ 50 escuadrar - ⊘ 16 APKT - ⊘ 18 APKT - ⊘ 30 APKT
	Aluminio y plásticos	- ⊘ 16 - ⊘ 18 - ⊘ 18 largo - ⊘ 30 - ⊘ 32 - ⊘ 50
Mandrinado	Indiferente	- Mandrino - Cabeza mandrinadora
Taladrado	Indiferente	- Portapinzas ⊘ 0-3 - Portabroca ⊘ 3-14 - Cono ⊘ >14 - Broca Plaquilla
Especiales	Indiferente	- Achaflado ⊘ 12 - Achaflado ⊘ 22 - Turbo
Fresado	Indiferente	- Portaherramienta Ni-Fresa - Portapinza-Fresa
Alineado	Indiferente	- Reloj comparador

Tabla 3.4: Relación de las operaciones, herramientas y materiales

Sensor	Unidad de medida	Precisión
Acelerómetro	Metros	0,005
Sensor de fase de corriente	Amperios	0,01
Sonda de temperatura	Grados centígrados	1
Detector de sonido	Decibelios	0,01

Tabla 3.5: Unidades de medida y precisión de los sensores

Todos los sensores realizan mediciones con una cadencia de un segundo, independientemente de la hora, operación realizada o estado de la máquina. Estas mediciones son almacenadas en una tabla de la base de datos que se describe en la próxima sección (3.3).

3.3. La base de datos

Desde marzo del 2019, los sensores han estado realizando mediciones y almacenando información. Para contener y estructurar toda la información medida, se dispone de una base de datos relacional basada en *SQL Server* denominada `CNC`. En esta base de datos se dispone de varias tablas con los registros del funcionamiento y las mediciones. De esas tablas interesan especialmente dos: `CNC_Funcionamiento` y `TiempoOperaciones`.

La tabla principal, denominada `CNC_Funcionamiento`, almacena la información recogida cada segundo por los sensores y se compone de los campos presentes en la *Tabla 3.6*.

Atributo	Introduce información	Tipo de dato y rango de valores	Notas
`Time_Stamp`	Sistema	Cadena de tiempo con formato: año-mes-día hora:mi:-seg	Fecha en la que se escribe la medición en la base de datos
`Time_Stamp_ms`	Sistema	Entero positivo	Tiempo en milisegundos que transcurren en la medición
`CNC_Acelerometrox`	Acelerometro-eje-x	Real	
`CNC_Acelerometroy`	Acelerometro-eje-y	Real	
`CNC_Acelerometroz`	Acelerometro-eje-z	Real	
`CNC_CorrienteFase1`	Sensor de fase de corriente 1	Real positivo	
`CNC_CorrienteFase2`	Sensor de fase de corriente 2	Real positivo	
`CNC_CorrienteFase3`	Sensor de fase de corriente 1	Real positivo	
`CNC_Temperatura`	sonda de temperatura	Entero	
`CNC_Sonido`	Detector de sonido	Real positivo	
`CNC_puerta`	Sensor de la puerta	0 ó 1	Puerta abierta (0) o cerrada (1)
`operacion_id`	Usuario	Entero entre 0 y 7	Código de la operación realizada
`material_id`	Usuario	Entero entre 0 y 5	Código del material utilizado
`orden-fab`	Usuario	Cadena de caracteres	Identificador de la pieza
`herramienta_id`	Usuario	Entero entre 0 y 31	Identificador de la herramienta

Tabla 3.6: Campos e información en la tabla `CNC_Funcionamiento`

Además de la información registrada por los sensores, el usuario operador de la fresadora introduce la información correspondiente a la operación, el material, la orden de fabricación y la herramienta que va a utilizar cuando comienza el trabajo con una pieza, a través de un programa de control instalado en un equipo con pantalla táctil. Este programa es el responsable de añadirlas en cada medición a la tabla que registra el funcionamiento.

Figura 3.1: Equipo con pantalla táctil con el sistema de control instalado
Fuente: Fundación CIDAUT

Por último, con el objetivo de detectar errores en las mediciones, se incluyó el campo `Time_Stamp_ms` que registra el tiempo (en milisegundos) que se requiere para la realización de las mediciones desde que se produce el evento que realiza la medición. Con ello, cuando el valor presentado sea superior a 1000 se considera que se ha "perdido" una medición y puede ser un indicativo de error en los sensores.

Paralelamente, se dispone de una segunda tabla, denominada `TiempoOperaciones`, en la cual se registra, de forma independiente a la tabla principal, el comienzo y fin de cada operación, junto al identificación de la operación, de la herramienta y la orden de fabricación.

El objetivo de esta segunda tabla es conocer el momento de comienzo y fin de operaciones para poder procesar más fácilmente el histórico de datos almacenados. También sirve para verificar la integridad de la información almacenada. Los campos de esta tabla están recogidos en *Tabla 3.7*.

La relación entre las dos tablas de uso se puede enunciar como "cada fila de la tabla `TiempoOperaciones` se relaciona con todas las filas de la tabla `CNC_Funcionamiento` cuya fecha de realización (`Time_Stamp`) este comprendido entre los valores de inicio (`inicioOp`) y fin (`finOp`) de la fila seleccionada".

Atributo	Introduce información	Tipo de dato y rango de valores	Notas
`Time_Stamp`	Sistema	Cadena de tiempo con formato: año-mes-día hora:mi:-seg	Fecha en la que se finaliza la acción. Actúa como identificador y clave ajena respecto de la tabla principal
`Time_Stamp_ms`	Sistema	Entero positivo	Tiempo en milisegundos que transcurren entre mediciones y escritura
`operacion_id`	Usuario	Entero entre 0 y 7	Código de la operación realizada
`herramienta_id`	Usuario	Entero entre 1 y 31	Identificador de la herramienta
`inicioOp`	Sistema	Cadena de caracteres con formato: día/mes/año hora:min:seg	Fecha de inicio de la operación.
`finOp`	Sistema	Cadena de caracteres con formato: día/mes/año hora:min:seg	Fecha de fin de la operación

Tabla 3.7: Campos e información de la tabla *tiempoOperaciones*

El siguiente modelo conceptual (*Figura 3.2*) describe las relaciones entre las entidades anteriormente descritas.

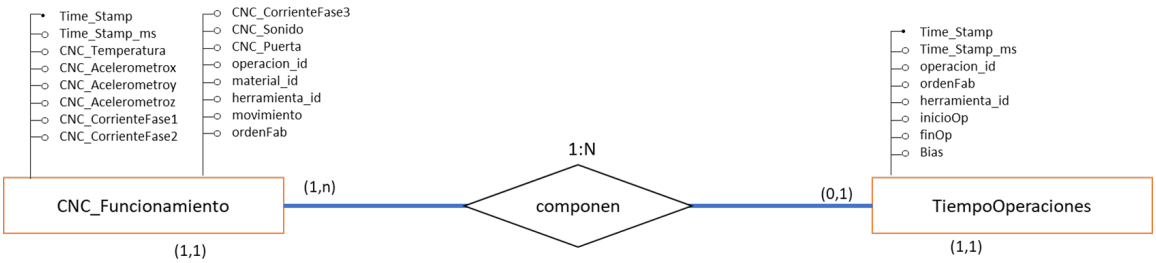


Figura 3.2: Diagrama entidad-relación de la base de datos, representando una imagen de esta antes de comenzar el proyecto

3.4. Resultados relevantes en actividades previas del proyecto iDIGIT4L

Como resultado de tareas de análisis realizadas anteriormente, se dispone de datos procedentes de los sensores de consumo de la máquina fresadora durante el proceso de arranque. En dicho proyecto se determinó que el arranque de la máquina constaba de cuatro fases diferenciadas según los valores dados por el sensor de fase de corriente del relé 1, y que se corresponden con la secuencia de acciones de puesta en marcha realizadas por el operario sobre la máquina fresadora.

Fase 1. El arranque comienza con un "pico" (consumo elevado de duración de unos segundos) de corriente, que se producen al pasar de consumo cero al consumo nominal.

Fase 2. Tras el pico inicial, durante un periodo de tiempo de entre 10 y 15 minutos aproximadamente, los valores de consumo permanecen constantes salvo las variaciones periódicas propias de la intensidad de corriente, tomando valores próximos entre 1,5 y 2 amperios. Esta fase se corresponde con el calentamiento de algunos componentes de la fresadora.

Fase 3. La fase dos finaliza con un segundo "pico" de corriente que se corresponde al arranque del PLC[1]. Este hecho se produce cuando el operador retira la seta de seguridad. En algunos casos, el consumo elevado de esta fase se puede extender hasta unos 30 segundos en el tiempo.

Fase 4. Una vez que se ha completado el arranque del PLC el valor de la intensidad de corriente vuelve a estabilizarse con valores próximos al valor nominal de la máquina (2,5 amperios)

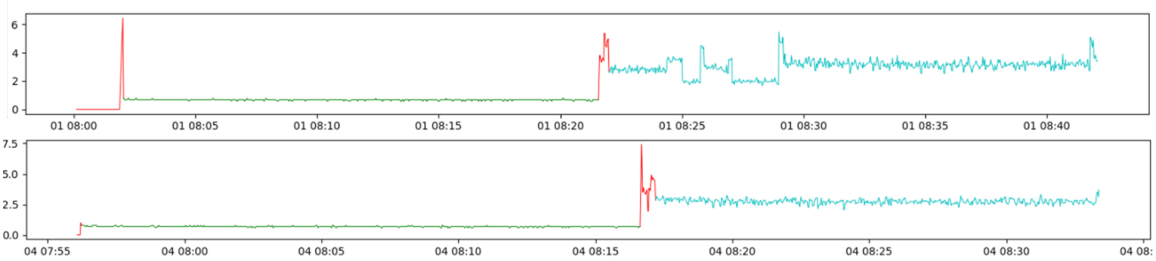


Figura 3.3: Representación de la corriente medida por el sensor de corriente de fase 1 durante el arranque de la máquina fresadora identificando sus fases anteriormente descritas.
Fuente: Fundación CIDAUT

[1] acrónimo del inglés *Programmable Logic Controller*, es una computadora digital industrial que ha sido reforzada y adaptada para el control de procesos de fabricación.

El comportamiento anterior describe lo que se denomina *modelo básico de arranque*. El principal problema que se presenta en el estudio de los arranques de la máquina fresadora es el desconocimiento de cuando realmente termina el arranque, pues la información disponible solo permite detectar como fin de arranque el momento en que comienza una operación. Por tanto, es difícil conocer exactamente cuando termina la fase 4 del arranque. Para solventar esta dificultad, se considera que un arranque finaliza cuando finaliza la fase 3, pues en dicho momento es posible trabajar con la máquina en las operaciones diarias.

Capítulo 4

Fundamento teórico

En este capítulo se van a presentar las técnicas utilizadas para estructurar y resolver el problema de la clasificación no supervisada de las actividades realizadas en la máquina fresadora.

Para estructurar y representar estas actividades se van a utilizar **series temporales**, y para la clasificación se recurre al algoritmo de **k-medias** (*k-means*). Por consiguiente, en este capítulo se tratará el concepto de serie temporal y ciertas transformaciones que se llevan a cabo para depurar la información recogida por la serie.

Por otro lado, se presenta el algoritmo de k-medias y sus fundamentos, con objeto de adaptar dicho algoritmo a su aplicación sobre series temporales. Es más, las series temporales que caracterizan las actividades presentan un comportamiento peculiar: un mismo "evento" puede tener diferentes momentos de ocurrencia y distinta duración. Esto se debe a las diferencias en el material con el que se trabaja y que no se registran. Así, la operación de planeado o contorneado dependen de la superficie y dimensiones del material sobre el que se trabaja o la operación de taladrado depende del grosor.

La solución de clasificación presentada, mediante la adaptación del algoritmo de k-medias, busca tener en cuenta las peculiaridades anteriores a la hora de clasificar las actividades estructuradas como series temporales.

4.1. Fundamentos sobre series temporales

En esta sección se presentan los conceptos de serie temporal, con las técnicas y modelos que se aplican en el análisis de las series temporales y que han sido aplicadas en el caso de estudio. Esta sección se basa en los capítulos primero y segundo de [2] y en los capítulos octavo y noveno de [3]. Las definiciones formales se recogen en la *Sección C.1*, y el estudio teórico detallado puede consultarse en [25].

Una **serie temporal** se define como la realización de un proceso estocástico (véanse las *Definiciones C.1 y C.2*) considerado el modelo de la serie. En el caso de estudio del presente trabajo, una serie temporal es considerada una sucesión de datos medidos en determinados momentos, ordenados cronológicamente y marcados por el momento de medición.

Estas mediciones pueden estar equiespaciadas en el tiempo o medidas a intervalos desiguales (véase *Definición C.3*). En el caso de estudio, las series temporales van a ser del primer tipo, con cadencia de un segundo, y por tanto, esta memoria se centrará exclusivamente en su estudio. Además, los algoritmos presentados e implementados son aplicables exclusivamente a este tipo de series.

Como además, las actividades realizadas tienen momento de inicio y fin definidos, entonces las series temporales del caso de estudio tienen longitud finita aunque variable. En consecuencia, se van a representar las series temporales con notación vectorial $\mathbf{X} = [x_1, \ldots, x_n]$ donde cada x_i denota la medición en el instante $i-$ésimo. En lo que resta de esta memoria se va a trabajar con este tipo de series temporales y con la notación introducida.

4.1.1. Componentes de una serie temporal: el modelo aditivo

El modelo básico y más general considera que una serie temporal se divide en tres componentes: tendencia, estacionalidad y residuo (*Definición C.4*).

La **tendencia** recoge el comportamiento regular de la serie y se identifica como los movimientos a largo plazo que sufre la media de los términos de la serie. La **estacionalidad** engloba las variaciones periódicas de la serie. Y el **residuo** agrupa los restos aleatorios que no pueden ser englobados en la tendencia o en la estacionalidad.

En esta memoria se trabaja bajo las hipótesis del **modelo aditivo** (*Definición C.7*), que supone que la serie es la suma de sus componentes. En el *Ejemplo C.8* se presenta la descomposición de una serie temporal según este modelo.

4.1.2. Series temporales estacionarias

Una serie temporal se dice **estacionaria** (*Definición C.9*) cuando es estable, es decir, la media y la varianza permanecen constantes a lo largo del tiempo.

Esto se da cuando los valores de los coeficientes de la serie oscilan entorno a un valor y su variabilidad respecto a dicho valor es constante. Es decir, gráficamente no se observan ni aumentos ni disminuciones sistemáticos en la representación continua de la serie.

Las propiedades de este tipo de serie permiten realizar, de forma sencilla, predicciones y validaciones apoyadas en las funciones de autocovarianza y autocorrelación (véase la pág. 132).

4.1.3. Determinar la tendencia de una serie: el algoritmo de medias móviles

Generalmente, para el análisis y comparación de series se requiere la eliminación de los efectos estacionarios. Para ello, existen diferentes técnicas para determinar la tendencia.

Estas técnicas dependen de si la tendencia es determinista (monótona en el tiempo) o evolutiva (con intervalos de crecimiento y decrecimiento) (*Definición C.12*). Para el primer caso se recurren a técnicas de regresión (consúltese la *Sección C.1.4* donde se recogen estas técnicas), y para las series temporales con tendencia evolutiva se recurre al algoritmo de medias móviles.

Algoritmo de Medias Móviles

Sea $\mathbf{X} = [x_1, \ldots, x_n]$ una serie temporal, se denota por $\mathbf{Y} = [y_1, \ldots, y_m]$ la serie temporal resultante.

1. Se define un "tamaño de ventana" denotado por $v \in \mathbb{N}$, que determina el número de observaciones que se contemplan para determinar una observación concreta. El valor v debe ser un natural impar en la versión centrada.

2. Dependiendo de la versión del algoritmo elegida:

   a) Si se consideran los momentos pasados y futuros *(versión centrada).*

   Sea por $u = \frac{v-1}{2}$, entonces para cada k natural en $[u, n-u]$ se define y_k como la media de las observaciones en el periodo $[k-u, k+u]$:

   $$y_k = \frac{1}{2u+1} \sum_{i=k-u}^{k+u} \omega_i \cdot x_i \qquad \text{donde} \qquad \frac{1}{2u+1} \sum_{i=k-u}^{k+u} \omega_i = 1 \tag{4.1}$$

   b) Si solo se consideran momentos pasados *(versión temporal).*

   Para cada k natural en $[v, n]$ se define y_k como la media de las observaciones en el periodo $[k-v, k]$:

   $$y_k = \frac{1}{v+1} \sum_{i=k-v}^{k} \omega_i \cdot x_i \qquad \text{donde} \qquad \frac{1}{v+1} \sum_{i=k-v}^{k} \omega_i = 1 \tag{4.2}$$

Con las notaciones anteriores, a v se lo denomina tamaño de ventana y a $\mathbf{Y}$ se lo denomina media móvil de $\mathbf{X}$. El orden del algoritmo es clave en el resultado final: si el orden es pequeño se suavizan los efectos de los residuos y si el orden es grande se suprimen tanto los residuos como la componente estacionaria. En general, se considera un orden pequeño aquel que no engloba una "oscilación" de la estacionalidad. La implementación se recoge en la *Sección B.1.*

4.2. El algoritmo de k-medias

4.2.1. Introducción

En esta sección se introducen las técnicas necesarias para encontrar patrones o "agrupamientos naturales" en un conjunto de datos de los cuales no se dispone de más información que el propio dato. Está basada en [13], donde se presenta la versión tradicional del algoritmo de k-medias entre otros.

Así pues, se trata de un caso de *aprendizaje no supervisado*, en el que se busca dividir un conjunto de datos, denotado por $\mathcal{D} = \{x_1, \dots x_n\}$, en diferentes subconjuntos, denominados **clústeres** denotados por $K_1, \dots K_k$, tales que, los elementos de cada clúster compartan características, desconocidas a priori, y que cada subconjunto difiera al máximo posible del resto.

A cada clúster se lo asigna un **centroide** (centro geométrico), también llamado *baricentro*, que actúa como representante de su clúster. Este centro geométrico viene determinado por una "aplicación" sobre los puntos que pertenecen cada clúster.

Para poder realizar esta clasificación se requiere disponer de una medida de similitud (generalmente una **distancia**), denotada por d, que permita comparar los objetos. Por tanto, un algoritmo de *clustering* busca minimizar las distancias entre elementos del mismo clúster y maximizar las distancias entre diferentes clústeres. Se denota por c_i al centroide del clúster K_i.

Las asignaciones a cada clúster se suelen representar mediante una matriz $W = (\omega_{i,j})_{1 \le i \le m; 1 \le j \le n}$ denominada **matriz de pertenencia** tal que:

$$\omega_{i,j} = \begin{cases} 1 & si \quad x_i \in K_j \\ 0 & si \quad x_i \notin K_j \end{cases}$$

Por otro lado, una clasificación de *clustering* es caracterizada y valorada mediante la distancia intra-clúster y la variación total intra-clúster:

- La **distancia intra-clúster** se define como:

$$\sum_{i=1}^{p} \sum_{\alpha_s \in K_i} \sum_{\alpha_r \in K_i}^{s} \frac{1}{2} d(\alpha_s, \alpha_r) = \sum_{i=1}^{p} \sum_{j=1}^{m} \sum_{l=j+1}^{m} \omega_{l,t} \cdot \omega_{j,t} \cdot d(x_j, x_l)$$

- La **variación intra-clúster de** K_t se denota por $S^t(W)$, y se define como la suma de las distancias entre cada elemento del clúster con su centroide. Esto es:

$$\mathcal{S}^t(W) = \sum_{x \in K_t} d(x, c_t) = \sum_{i=1}^{m} \omega_{i,t} \cdot d(x_i, c_t)$$

  Análogamente, se considera la **variación total intra-clúster**, y se denota por $\mathcal{S}(W)$ como la suma de las variaciones de cada clúster. Esto es:

$$\mathcal{S}(W) = \sum_{t=1}^{n} \mathcal{S}^t(W) = \sum_{t=1}^{n} \sum_{i=1}^{m} \omega_{i,t} \cdot d(x_i, c_t)$$

4.2.2. Algoritmo de k-medias versión tradicional

El algoritmo de *clustering* por excelencia es el **algoritmo de k-medias** (*k-means*), también denominado *algoritmo de Lloyd*, pues fue propuesto por primera vez por Stuart Lloyd en 1957.

Este algoritmo, asigna un dato $a \in \mathcal{D}$ al clúster K_i si y solo si:

$$d(a, c_i) \leq d(a, c_j) \quad \forall j = 1 \dots n \text{ con } i \neq j$$

Una vez asignados los datos a un único clúster, se recalculan los centroides como la media[1] de los datos que pertenecen a dicho clúster. El procedimiento anterior es repetido hasta que se cumple cierto criterio de parada. Los criterios de parada más usuales son:

- La asignación de clústeres permanece constante tras una iteración.
- La variación total intra-clúster es inferior a cierto valor de tolerancia.

Algoritmo de k-medias *(versión tradicional)*

Sea k el número de clústeres predefinidos, sean $K_1, .., K_k$ los clústeres y $c_1, .., c_k$ los centroides asociados a cada clúster. Sea $\mathcal{D} = \{x_1, .., x_m\}$ el conjunto de datos.

1. Inicialización: se inicializan aleatoriamente cada centroide.
2. Asignación: se realiza la asignación de clústeres para cada $x \in \mathcal{D}$, de forma que x pertenecerá al clúster cuyo centroide minimice su distancia respecto al conjunto de centroides.

$$x \in K_i \Leftrightarrow i = \arg\min_{j=1..k}(d(x, c_j)) \tag{4.3}$$

3. Calculo de centroides: se reajustan los centroides como la media de los datos de cada clúster.

$$c_i = \frac{1}{|K_i|} \sum_{x \in K_i} x \tag{4.4}$$

4. Evaluación: Se comprueba el criterio de parada, que se verifica cuando la asignación de clústeres permanece constante entre dos iteraciones consecutivas.
   Si no se cumple este criterio, se repiten los pasos 2, 3, y 4 con los nuevos centroides.

[1] Entendiendo por media de un conjunto de vectores dato el vector cuya componente $i-ésima$ es la media de la componente $i-ésima$ de cada vector dato

El algoritmo de k-medias requiere que se determine previamente el número de clústeres en los que separar el espacio. Para ello se suelen emplear técnicas heurísticas, pues la elección adecuada del número de clústeres puede afectar notablemente al resultado. También es necesario inicializar los centroides en la primera iteración del algoritmo, para lo que generalmente se recurre a una asignación aleatoria.

El algoritmo de k-medias es un problema de complejidad *np-hard* siempre que la dimensionalidad de los datos sea mayor o igual que dos, es decir, el coste computacional (y por tanto el tiempo necesario) crecen de forma polinómica respecto a un crecimiento lineal del número de datos[2].

La implementación en pseudocódigo se recoge la *Sección B.6* y se realiza para la versión de la matriz de pertenencia del algoritmo. En la *Sección D.3* se presenta el algoritmo para la matriz de pertenencia.

Finalmente, para poder aplicar el algoritmo de k-medias sobre series temporales, se requiere de dos elementos clave: una medida de comparación y una función promedio. En las siguientes secciones se estudian estas funciones, buscando que sean adecuadas a las peculiaridades de las series temporales representativas de actividades descritas en la introducción de este capítulo.

4.3. Medida de comparación de series temporales: La DTW

En esta sección se busca definir una medida de comparación para comparar dos series temporales de distinta duración y adecuada a las peculiaridades del caso de estudio. Para ello, se introduce el concepto de **deformación en el tiempo** como relación fundamental entre dos series temporales y se define una "métrica" adecuada.

Esta sección está estructurada según [7], siendo esta referencia bibliográfica introductoria completada mediante [5], [6], [8], [9]. Los resultados han sido adaptados a su aplicación sobre series temporales reales de longitud finita. Las demostraciones y el estudio en más detalle se pueden consultar en [25].

Como notación general, sean $\mathbf{X} = [x_1, \ldots, x_n]$ e $\mathbf{Y} = [y_1, \ldots, y_m]$ dos series temporales, se suele comparar una serie temporal frente a la otra. Si se compara $\mathbf{X}$ con respecto a $\mathbf{Y}$ entonces a $\mathbf{X}$ se le denomina serie **muestra** y a $\mathbf{Y}$ se le denomina serie **prueba**.

[2]Esta afirmación se prueba en [15]

4.3.1. La problemática en la comparación de series temporales. Deformaciones en el tiempo

La primera aproximación en la comparación de dos series temporales consiste en considerar las series como vectores y aplicar la distancia euclídea (o cualquier otra distancia vectorial). Es decir ,sean $\mathbf{X} = [x_1, \ldots, x_n]$ e $\mathbf{Y} = [y_1, \ldots, y_n]$ dos series temporales de la misma longitud entonces

$$d_E(\mathbf{X}, \mathbf{Y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{4.5}$$

Esto se conoce como *comparación según el orden cronológico*. Esta idea, aunque sencilla, resulta bastante incompleta pues no permite comparar series temporales de distinta longitud, ni es capaz de detectar el efecto *time warping* o **deformación en el tiempo**. El efecto de deformación en el tiempo se produce cuando se realiza una misma acción, pero a diferentes velocidades o en diferentes momentos de ocurrencia.

Por ejemplo, supongamos dos personas que realizan el mismo recorrido y que cada minuto anotan la altitud a la que se encuentran. Además, en la parte más alta una de ellas se desplaza el doble de rápido que la otra persona. Las series temporales podrían ser de la forma:

$$\begin{aligned} \mathbf{X} &= [1, 2, 3, 3, 2, 1] && \text{(persona rápida)} \\ \mathbf{Y} &= [1, 2, 3, 3, 3, 3, 2, 1] && \text{(persona lenta)} \end{aligned}$$

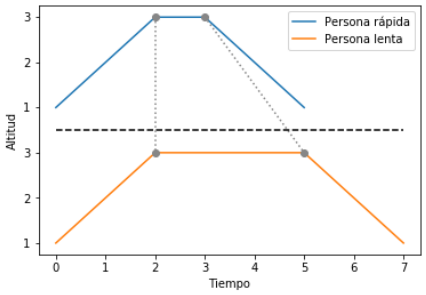


Figura 4.1: Ejemplo del efecto de deformación en el tiempo

El efecto de deformación en el tiempo se observa en la parte más alta (véase *Figura 4.1*), pues ambas personas realizan el mismo recorrido, pero como una de ellas es más rápida en ese tramo la altitud de 3 unidades aparece menos veces recogida.

La distancia euclídea es incapaz de comparar ambas series temporales. Se requiere que la medida de comparación sobre series temporales usada sea capaz de detectar las series del ejemplo como equivalentes, pues esta es la peculiaridad de las actividades medidas. En la *Figura 4.1* se puede observar (en gris con línea punteada) como se relacionan los puntos claves de ambos

recorridos, aunque no coincidan en el tiempo, pero se corresponden al mismo punto de longitud recorrida. En el ejemplo, el objetivo perseguido se entiende como "la persona rápida espera a la persona lenta antes de iniciar el descenso".

El ejemplo ilustra la relación fundamental entre series temporales, la **extensión por deformación** (*Definición C.18*). Se dice que una serie $\mathbf{Y}$ es **extensión** de una serie $\mathbf{X}$ si $\mathbf{Y}$ es el resultado de duplicar consecutivamente (respetando el orden cronológico) algunos términos de $\mathbf{X}$. De forma análoga, $\mathbf{Y}$ es una condensación de $\mathbf{X}$ si $\mathbf{Y}$ es el resultado de suprimir algún término de $\mathbf{X}$ duplicado consecutivamente.

Por ejemplo, considerando la serie $[1, 1, 2, 3]$ una extensión es $[1, 1, 2, 2, 3]$ y una condensación es $[1, 2, 3]$. Además, la serie $[1, 2, 2, 3]$ también es una deformación en el tiempo. La medida de comparación considerada debe marcar estas series como "iguales" (es decir, su "distancia" es cero).

4.3.2. La distancia DTW

Para poder comparar series de distinta longitud y detectar los efectos de las deformaciones en el tiempo se va a construir un nuevo orden cronológico en el que comparar las series temporales. Así, aparece el concepto de **camino de deformación** (*Definición C.13*).

Dadas dos series $\mathbf{X} = [x_1, \ldots, x_n]$ e $\mathbf{Y} = [y_1, \ldots, y_n]$ muestra y prueba respectivamente, un camino de deformación de orden $n \times m$ y cardinal P es una sucesión finita de puntos bidimensionales

$$\Psi = \{(i_1, j_1), \ldots, (i_P, j_P)\} \subset \{1, \ldots, n\} \times \{1, \ldots, m\}$$

que satisfacen ciertas restricciones temporales, y establecen un nuevo orden cronológico de comparación $1, \ldots, P$. Con abuso de notación, se denota $(i_s, j_s) = \Phi(s) = (\phi_X(s), \phi_Y(s))$.

Entonces, comparando ambas series con el nuevo orden cronológico se obtiene el coste del camino de deformación. Esto es:

$$C_{\mathbf{X},\mathbf{Y}}(\Psi) = \sum_{(i,j)\in\Psi} d(x_i, y_j) = \sum_{i=1}^{P} d(x_{\Psi_X(i)}, y_{\Psi_Y(j)})$$

Considerando todos los posibles caminos, se define la **curva de deformación** como el camino de menor coste. Y la distancia DTW (*Dynamic Time Warping*) como el coste asociado a la curva de deformación (*Definición C.15*).

La distancia DTW es capaz de ignorar las deformaciones en el tiempo, estableciendo asociaciones entre los índices de la serie muestra y prueba sin importar el momento de ocurrencia (*Teorema C.19*). Sin embargo, el cálculo como un mínimo hace el problema inabarcable cuando las longitudes de las series son lo suficientemente grandes. Por ello, el cálculo efectivo de la DTW y de la curva de deformación se realiza mediante la matriz DTW.

4.3.3. Algoritmo para el cálculo de la distancia DTW: la matriz DTW

En esta sección, se presenta un algoritmo capaz de calcular la distancia DTW de dos series con complejidad cuadrática, mediante la adecuada construcción de una matriz (basado en [7]).

El algoritmo *Dynamic Time Warping* (DTW) *(versión primera)*

Sean $\mathbf{X} = [x_1, \ldots, x_n]$ e $\mathbf{Y} = [y_1, \ldots, y_m]$ dos series temporales cualesquiera de longitud n y m respectivamente. Se denota por $A_{DTW} = (a_{i,j})_{i=1\ldots n; j=1\ldots m}$ la matriz de dimensiones $n \times m$ del algoritmo DTW. Además, sea d una distancia sobre espacio en el cual están definidos los valores de x_i e y_i $\forall i$.

1. Lo primero es construir la matriz DTW. Para ello se sigue el siguiente procedimiento.

   1.1. Se define el primer elemento de la matriz DTW (A_{DTW}) como la distancia entre x_1 e y_1.

   $$a_{1,1} = d(x_1, y_1) \tag{4.6}$$

   1.2. Se calcula la primera fila y columna de la matriz DTW.
   Para cada $j = 2 \ldots m$ se define

   $$a_{1,j} = d(x_1, y_j) + a_{1,j-1} \tag{4.7}$$

   Y para cada $i = 2 \ldots m$ se define

   $$a_{i,1} = d(x_i, y_1) + a_{i-1,1} \tag{4.8}$$

   1.3. Para cada $i = 2 \ldots n$ y para cada $j = 2 \ldots m$ se define el elemento $a_{i,j}$ como:

   $$a_{i,j} = d(x_i, y_j) + \text{mín}(a_{i,j-1}, a_{i-1,j}, a_{i-1,j-1}) \tag{4.9}$$

2. La distancia DTW entre ambas series es el término $a_{n,m}$.

El objeto principal para el trabajo con la distancia DTW es la matriz DTW, pues permite el cálculo tanto del valor $d_{DTW}(\mathbf{X}, \mathbf{Y})$ como de la curva de deformación (véase *Sección 4.3.4*). Además, dicha matriz determina una relación entre los índices de cada una de las series, como se recoge en la *Definición C.21*. La implementación en pseudocódigo se encuentra en la *Sección B.2*.

La justificación de que distancia DTW entre la serie muestra y prueba es el término (n, m)–ésimo de la matriz DTW se presenta en la *Sección C.2.4*, así como las restricciones que se deben verificar para que el algoritmo sea completo. Dichas restricciones, redundantes en la versión primera del algoritmo DTW, son necesarias en las otras versiones del algoritmo.

El algoritmo *DTW* expuesto anteriormente reduce la complejidad del problema, pues basta con construir adecuadamente la matriz DTW. Así, si la longitud de las series temporales muestra y prueba son n y m respectivamente, la complejidad del algoritmo anterior es de $\mathcal{O}(nm)$ (coste de construcción de una matriz (n,m)–dimensional). Por tanto, suponiendo $n \geq m$ la complejidad es de $\mathcal{O}(n^2)$.

4.3.4. La curva de deformación y las asociaciones entre índices

Además del valor de la distancia DTW interesa especialmente la curva de deformación, pues aporta gran cantidad de información útil y necesaria en la comparación de las series temporales. En esta sección se presenta un algoritmo capaz de hallar la curva de deformación a partir de la matriz DTW (basado en [7]).

Algoritmo para hallar la curva de deformación

Sea $A_{DTW} = (a_{i,j})_{1\leq i\leq n, 1\leq j\leq m}$ la matriz obtenida por el algoritmo DTW.

1. Se comienza con el par de índices $\delta(0) = (n,m)$.
2. Se define $\delta(1)$ como
$$\delta(1) = \underset{(\alpha,\beta)}{\arg\min}(a_{\alpha,\beta} : \ \alpha \in \{n, n-1\} \text{ y } \beta \in \{m, m-1\})$$
3. Recurrentemente, para cada $k = 1...(P-1)$ se define
$$\delta(k) = \underset{(\alpha,\beta)}{\arg\min}(a_{\alpha,\beta} : \ \alpha \in \{n-k-1, n-k\} \text{ y } \beta \in \{m-k-1, m-k\})$$
donde se supone que $\delta(P-1) = (1,1)$
4. Entonces $\Phi(k) = \delta(P-k), \quad k = 1 \dots P$

Este algoritmo, se fundamenta en el *Teorema C.22*, y su implementación se recoge en la *Sección B.3* con una construcción de forma iterativa. Además, la curva de deformación marca la comparación entre ambas series mediante extensiones y condensaciones sobre la serie prueba (véase *Sección C.2.5*).

Por otro lado, interesa conocer para cada índice de la serie muestra i los índices de la serie prueba $j_1, \dots j_s$ que $(i, j_t) \in \Phi$. Esto se denomina **asociación** de índices (véase *Sección C.2.6*), y establecen cuando un mismo evento sucede a distintas velocidades o tiene distinto momento de inicio.

La técnica para determinar γ para cada s es recorrer de forma ordenada la curva de deformación observando la primera componente de cada par de índices. La implementación es

pseudocódigo recogida en la *Sección B.4* sigue esta idea. Como notación, $\gamma(s)$ representa al conjunto de índices de la serie prueba que se asocian con el índice s de la serie muestra.

Para concluir esta sección, cabe destacar el ejemplo de aplicación de estos algoritmos recogido en la *Sección C.2.7*. Además, existen diferentes versiones del algoritmo DTW, recogidas en la *Sección C.2.8*. Estas versiones se caracterizan por modificar tres aspectos del algoritmo:

- Cambio de los patrones de recursividad y del coste. Se basa en ponderar con diferente peso las coincidencias sobre las deformaciones. En la versión presentada en la *Sección C.2.8 apartado 1* se penalizan las coincidencias en favor de las deformaciones.
- Aplicación de una restricción de localidad. Las restricciones de localidad impiden deformaciones temporales superior a cierta longitud. En estos casos cobra una gran importancia el cumplimiento de las restricciones enunciadas en la *Sección C.2.4*.
- Distancia p-DTW. Al igual que con la distancia vectorial p, se cambia la definición de coste de deformación (*Definición C.26*). En el caso de la librería `tslearn` se usa la 2–DTW, cuyo coste para el camino Ψ respecto de las series $\mathbf{X}$ e $\mathbf{Y}$ viene dado por

  $$C^2_{\mathbf{X},\mathbf{Y}}(\Psi) = \left(\sum_{(i,j)\in\Psi} d(x_i, y_j)^2 \right)^{1/2}$$

  En la *Sección C.2.8 apartado 3* se desarrolla esta versión de la DTW y su relación con la *versión primera*.

4.4. Función promedio. Media de Fréchet DBA

Dado que se comparan series temporales de diferente longitud, la media de las series como media aritmética cada componente carece de sentido. En esta sección se busca definir una "media" aplicable a series temporales, para lo que se recurre a la *media de Fréchet*.

Se denota por $\mathcal{T}$ al conjunto de las series temporales de longitud finita y variable, y sea $\mathcal{D} = \{\mathbf{X}_1, ..., \mathbf{X}_m\} \subset \mathcal{T}$. Se define la **función de Fréchet** bajo la DTW sobre el conjunto $\mathcal{D}$ y para la serie $\mathbf{Z}$ como

$$F_{\mathcal{D}}(\mathbf{Z}) = \sum_{i=1}^{m} d_{DTW}(\mathbf{Z}, \mathbf{X}_i)$$

La **media de Fréchet** μ como la serie temporal que hace mínimo el valor de la función de Fréchet para $\mathcal{D}$

$$\mu = \underset{\mathbf{M}\in\mathcal{T}}{\arg\min}\ F_{\mathcal{D}}(\mathbf{M}) \tag{4.10}$$

De forma análoga a la DTW, el cálculo de la media de Fréchet por la definición es muy costoso. Para ello se utiliza el algoritmo DBA (*DTW Barycenter Average*) que se apoya en las asociaciones de índices de la DTW y limita la búsqueda de la serie promedio a una duración fija.

Algoritmo DBA

Sea $\mathcal{D} = \{\mathbf{X}_1, \ldots, \mathbf{X}_m\}$ el conjunto con las secuencias a promediar, con n_i la longitud de la serie $\mathbf{X}_i = [x_1^i, \ldots, x_{n_i}^i]$.

1. Se supone que existe una serie temporal $\mathbf{C} = (c_1, \ldots, c_t)$ sobre la que se realiza la comparación.
2. Se toma T como la longitud de $\mathbf{C}$.
3. Sea $\gamma_i(s)$ el conjunto de los índices de la serie $\mathbf{X}_i$ asociados al índice s de la serie $\mathbf{C}$ por la distancia DTW. Sea $\mathbf{C}' = [c'_1, \ldots, c'_t]$ la media de Fréchet, entonces dado $s \in \{1, ..., t\}$ se define

$$c'_s = \frac{1}{\sum_{i=1}^{n} |\gamma_i(s)|} \left(\sum_{i=1}^{n} \sum_{k \in \gamma_i(s)} x_k^i \right) \tag{4.11}$$

   Donde $|\gamma_i(s)|$ denota el número de elementos del conjunto $\gamma_i(s)$

Es decir, el algoritmo DBA define una "media" entre series temporales respecto de $\mathbf{C}$ como la serie temporal cuya componente $s - ésima$ es dada por la media de las componentes de las series temporales a promediar cuyos índices se asocian con s.

La complejidad del algoritmo DBA para m series temporales suponiendo conocida la distancia DTW y las asociaciones es lineal. Si se incluye el tiempo necesario para el cálculo de la distancia DTW, suponiendo que la longitud del centroide es T y que la longitud máxima de la serie es n entonces la complejidad temporal es de $\mathcal{O}(n \cdot m \cdot T)$. Generalmente, n es el mayor de los tres valores y por tanto la complejidad es $\mathcal{O}(n^3)$. Además, la serie $\mathbf{C}$ es dada por el centroide de la iteración anterior del algoritmo de k-medias. En la *Sección D.4* se recogen los detalles clave del algoritmo DBA y se ilustra en dos ejemplos.

4.5. Algoritmo de k-medias aplicado a series temporales

La aplicación del algoritmo de k-medias a series temporales es trivial, una vez que se usa la DTW como medida de comparación y el algoritmo DBA como función promedio.

En su implementación, dado el elevado coste de la DTW, y teniendo en cuenta que las asociaciones de índices derivadas se requieren para el algoritmo DBA, el hecho de calcular una única vez la DTW y almacenar la matriz obtenida hasta la iteración siguiente del algoritmo supone una gran reducción en el coste. Los detalles de la adaptación se pueden consultar en la *Sección D.4.1* y la implementación en la *Sección B.7*.

4.6. Los diferentes enfoques a la clasificación de series temporales

En esta sección se desarrollan los principales enfoques para resolver el problema de clasificar un conjunto de series temporales. Está basado en las ideas desarrolladas en [15].

En particular, existen tres enfoques que difieren en la representación dadas a las series temporales previa a su clasificación.

- **Conversión discreta.** Para cada serie temporal se extraen un conjunto de parámetros o características discretas (en forma de vector) y se aplica el algoritmo de clasificación (k-medias tradicional) sobre los parámetros. Cuando el modelo de las series temporales es conocido, se pueden tomar los parámetros de dicho modelo. Dado que esto no es lo más usual, se suelen recurrir a caracterizaciones estadísticas (media, varianza, etc.) de la serie, así como a otras propiedades inherentes (longitud, cadencia máxima de medición, etc.).

- **Clasificación basada en la forma.** En este caso, se clasifica según la propia serie temporal, aplicando el algoritmo de k-medias bajo la distancia DTW como se presenta en las secciones anteriores.
  Una variante de este enfoque consiste se aplica algún filtro o transformación a las series temporales antes de realizar la clasificación. Las transformaciones más usuales son la eliminación de los efectos estacionales (generalmente sin eliminar las variaciones cíclicas cuando existan) o la segmentación según el modelo.

- **Enfoque híbrido**. Combina los dos enfoques anteriores, realizando en primer lugar una división del conjunto de datos por conversión discreta de las series temporales y en segundo lugar aplica un algoritmo de clasificación basada en la forma. Sea $\mathcal{D}$ el conjunto de las series temporales dato a clasificar, se distinguen dos fases:

  1) Se realiza una descomposición discreta de cada serie temporal, y se extraen un conjunto de parámetros de cada serie. Con dichos parámetros , se descompone $\mathcal{D} = D_1 \sqcup \ldots \sqcup D_n$.
  2) Para cada uno de los subconjuntos D_i se aplica un algoritmo de clasificación sobre series temporales basada en la forma. Con ello se tiene una segunda clasificación de las series temporales.

$$\begin{aligned} D_i &= K_{i,1} \sqcup \ldots \sqcup K_{i,m} \\ \mathcal{D} &= \bigsqcup_{i=1}^{n} \bigsqcup_{j=1}^{m} K_{i,j} \end{aligned} \tag{4.12}$$

Generalmente, el coste computacional y la complejidad de un algoritmo de clasificación basado en la forma suele ser superior al coste de un algoritmo basado en la conversión discreta. Por tanto, el enfoque híbrido presenta una importante ventaja, pues sin renunciar a la información que aporta la propia serie, reduce el coste al dividir el conjunto total de datos sobre el que aplica la clasificación basada en la forma. Por ejemplo, dado que la DTW tiene complejidad cuadrática, el uso de k-medias bajo la DTW tiene una mayor complejidad que el uso de k-medias bajo una distancia vectorial.

Capítulo 5

Estudio del problema y definición de un Modelo

En este capítulo se recogen los detalles del estudio inicial sobre la máquina fresadora y se define una arquitectura de modelo de clasificación. Dicha arquitectura se basa en las peculiaridades de las series temporales que modelan las actividades realizadas (un mismo evento puede suceder en diferentes momentos temporales con distinta duración), y busca minimizar las debilidades de la DTW.

Los detalles de la implementación del modelo y del Sistema de Detección de Anomalías en *Python* se desarrollan en los *Capítulos 7 y 8* y se justifica en este capítulo.

En primer lugar, se elige la forma de representar la información y se realiza un análisis de los datos extraídos. Como resultado, se diseña la arquitectura del modelo de clasificación, que requiere de ciertos parámetros propios, que se determinan en el *Capítulo 9*. En dicho capítulo, también se realiza una valoración del modelo construido.

5.1. Análisis de la información

En esta sección se desarrollan los detalles relacionados con la representación de la información y las transformaciones adecuadas para el limpiado de los datos. Se completa con un estudio y análisis que justifica la elección del Modelo de Clasificación. Este estudio se va a realizar sobre la información recolectada entre el 1 mayo de 2019 y el 30 de mayo de 2020[1].

Como se detalla en las *Secciones 3.2 y 3.3* , los sensores no intrusivos realizan mediciones con cadencia de un segundo, siendo cada medición etiquetada con la fecha exacta de realización, el tipo de operación que se realizaba, el material sobre el que se trabajaba y la herramienta utilizada.

[1]Debido a la situación producida por el Covid-19, la máquina fresadora ha sido destinada a otro tipo de producciones, y a partir de marzo del 2020 solo se han registrado la información relativa a los arranques.

Un primer estudio de la precisión sobre los datos recogidos y de los conocimientos adquiridos en el proyecto IDIGIT4L, indica que información recogida por el sensor de fase de corriente 1 (denominada *corriente de fase* en esta memoria) es la más adecuada para representar las actividades. Así, cada actividad realizada es representada mediante una serie temporal, dada por las mediciones del sensor de fase de corriente 1. Por otro lado, se requiere discriminar las actividades por el tipo de operación realizada y por el material utilizado. La *Figura 5.1* muestra un ejemplo de representación de una realización de la operación de contorneado sobre acero.

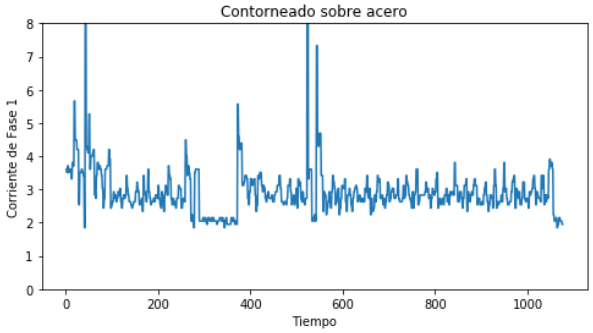


Figura 5.1: Representación de una operación de contorneado sobre el acero

Para diferenciar cada tipo de operación con cada tipo de material, se define un identificador único denominado *opermat*. Utilizando los identificadores numéricos de las *Tablas 3.1 y 3.2* para el tipo de operación y material, este identificador se define como diez veces el número asociado al tipo de operación más el número asociado al material. Por ejemplo, la operación anterior, de contorneado (operación 3) sobre acero (material 3), se le asocia el valor *opermat* de 33. Este identificador es el que se utiliza en las siguientes figuras.

El primer lugar, se desea conocer aquellos tipos de actividad (*opermats*) de los que se disponga de suficientes realizaciones para que los modelos construidos sean precisos y estables. En la *Figura 5.2* se representan el número de realizaciones en el periodo de tiempo estudiado.

Por tanto, el material más utilizado es el acero (código 3), que destaca notablemente para cada tipo de operación. Así, el estudio se centrará en cada tipo de operación realizada sobre acero y en el proceso de arranque de la máquina. En el *Apéndice E* se recoge la representación de ciertas actividades de los tipos de interés elegidas específicamente para ilustrar su comportamiento y su variedad.

Dado que la principal debilidad de la DTW son las deformaciones temporales excesivas, junto con la gran variedad de duración en las actividades del caso de estudio, se desea acotar las realizaciones utilizadas para entrenar el modelo según su duración. Además, esto permite eliminar actividades puntuales o incompletas que afectarían a la precisión del modelo. La siguiente figura (*5.3*) recoge las actividades agrupadas según su duración.

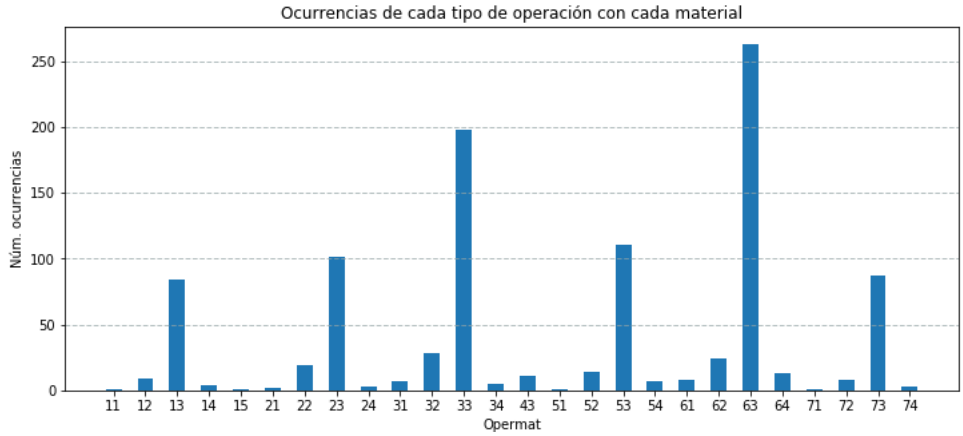


Figura 5.2: Ocurrencias de cada tipo de operación con cada material

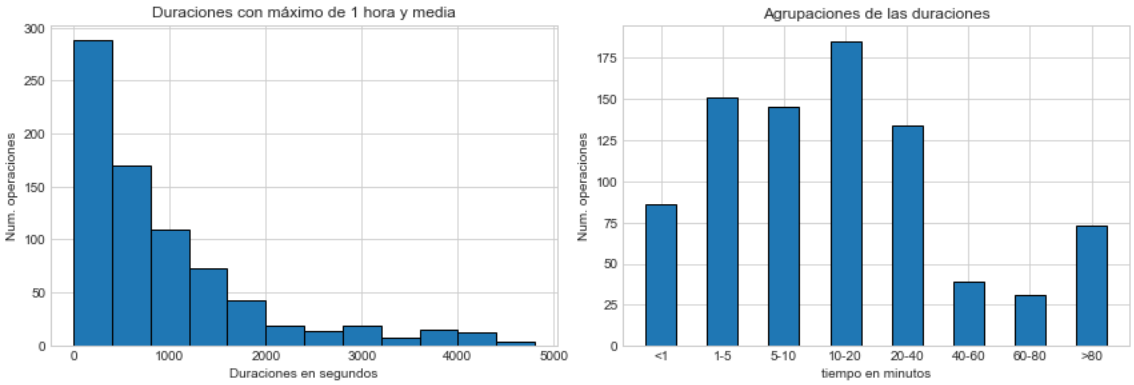


Figura 5.3: Histograma de las duraciones de las operaciones sobre el acero con duración inferior a una hora y 30 minutos (izquierda) y diagrama de barras con las duraciones agrupadas (derecha)

Para interpretar la variedad en la duración de las operaciones se recurre a la experiencia del operario encargado de la máquina fresadora:

- Las operaciones de poca duración (inferiores a cinco minutos) son asociadas al "repaso" o complemento de una pieza.
- Las operaciones de mucha duración (superiores a una hora) suelen ser debidas a piezas peculiares o de especial complejidad, acciones de revisado sobre la máquina fresadora y a errores humanos en el etiquetado.

Por lo tanto, para no malograr la eficacia y validez del Modelo de Clasificación, se considerarán actividades válidas aquellas cuya duración se encuentre entre 5 minutos y una hora.

Imponiendo estas restricciones, el número de operaciones válidas para el modelo se recogen en la *Figura 5.4*. En esta figura, se puede observar que las restricciones sobre la duración pueden ser demasiado exigentes para las operaciones identificadas por los códigos 13 y 73 de *opermat*. Sin embargo, para no perder generalidad y preservar condiciones homogéneas en cada operación, se mantienen dichas restricciones.

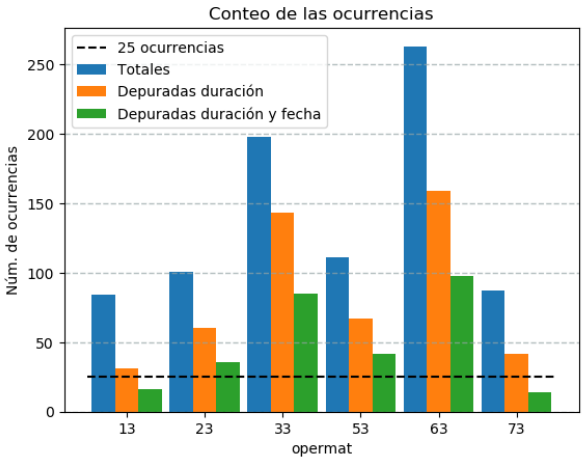


Figura 5.4: Diagrama de barras del número de operaciones resultantes: en azul sin imponer ningún tipo de restricción, en naranja tras exigir que la duración sea mayor de 5 minutos y menor de una hora y en verde tras exigir también que sean posteriores al 1 de o de 2019

El otro tipo de actividad que se realiza sobre la máquina lo comprende el proceso de arranque (descrito en la *Sección 3.4*). Para este caso, se repite el mismo análisis que se realizó sobre las operaciones.

De forma análoga, contrastando la variedad de las duraciones con la opinión fundada del operario de la máquina fresadora, se caracterizan los arranques con una duración de menos de tres minutos como actividades de mantenimiento u otro tipo de necesidades puntuales. Por tanto, se deben omitir del conjunto de datos que se utiliza en el entrenamiento del modelo.

Además, como se puede observar en la *Figura 5.1*, la corriente de fase tiene un importante comportamiento estacionario, característico de la corriente alterna, que altera los resultados y puede aumentar el coste computacional. Bajo las hipótesis del modelo aditivo, se descompone la serie y se elimina la estacionalidad aplicando el algoritmo de medias móviles en su versión centrada y con un tamaño de ventana, fijado experimentalmente, en 10. Con esto, se mantienen de las series su tendencia y su residuo. Los resultados de este tratamiento se representan en la *Figura 5.6*. En lo que resta, se supone que dicho tratamiento ha sido aplicado sobre las series de estudio.

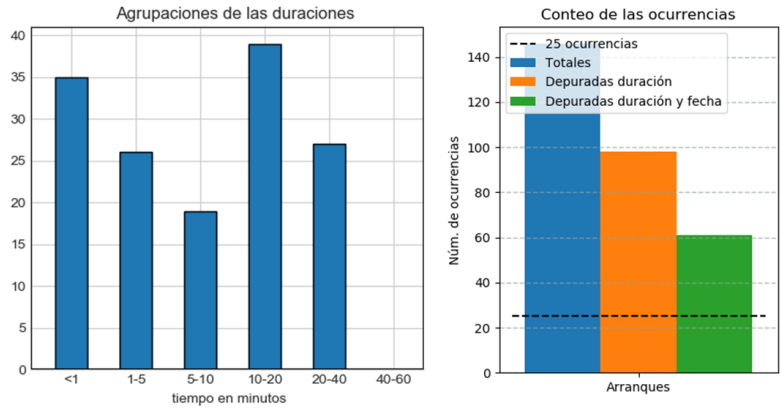


Figura 5.5: Agrupaciones de los arranques según su duración (izquierda) y diagrama de barras del número de operaciones resultantes (derecha): en azul sin imponer ningún tipo de restricción, en naranja tras exigir que la duración sea mayor de 3 minutos y menor de una hora y en verde tras exigir también que sean posteriores al 1 de septiembre de 2019

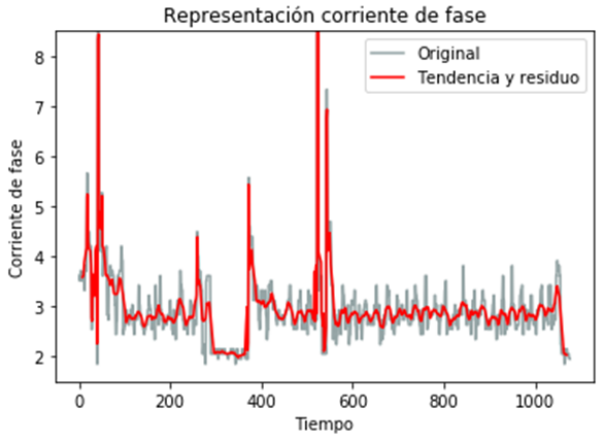


Figura 5.6: En gris la serie temporal original y en rojo la serie con la estacionalidad suprimida por el algoritmo de medias móviles con tamaño de ventana 10 en la versión centrada.

5.2. Hipótesis de la estacionalidad sobre las series temporales

En esta sección se busca validar si la hipótesis "las series temporales que representan las actividades tienen modelo estacionario" se puede asumir como cierta o debe ser descartada.

Para ello se va a realizar la **Prueba de Dickley-Fuller**[2] (también denominada **test ADF**) mediante el criterio de contraste de hipótesis del p-valor. En este caso, utilizaremos la Prueba de Dickley-Fuller aumentada con las siguientes hipótesis:

- **Hipótesis nula (H0)**: la serie temporal posee una raíz unitaria[3]. En este caso la serie es no estacionaria.
- **Hipótesis alternativa (Ha)**: la serie temporal no tiene una raíz unitaria, por lo que es estacionaria.

La prueba del p-valor rechaza la hipótesis nula cuando dicho valor es inferior a 0,05, y la acepta en caso contrario.

Para realizar esta prueba, utilizaremos la función `addfuller` del módulo *Python* `statsmodels.tsa.stattools`. Esta función calcula el p-valor. Los resultados obtenidos se pueden consultar en la *Figura 5.7*. En dicha figura se puede observar como la hipótesis de estacionalidad puede ser aceptada para las operaciones y debe ser rechazada para los arranques.

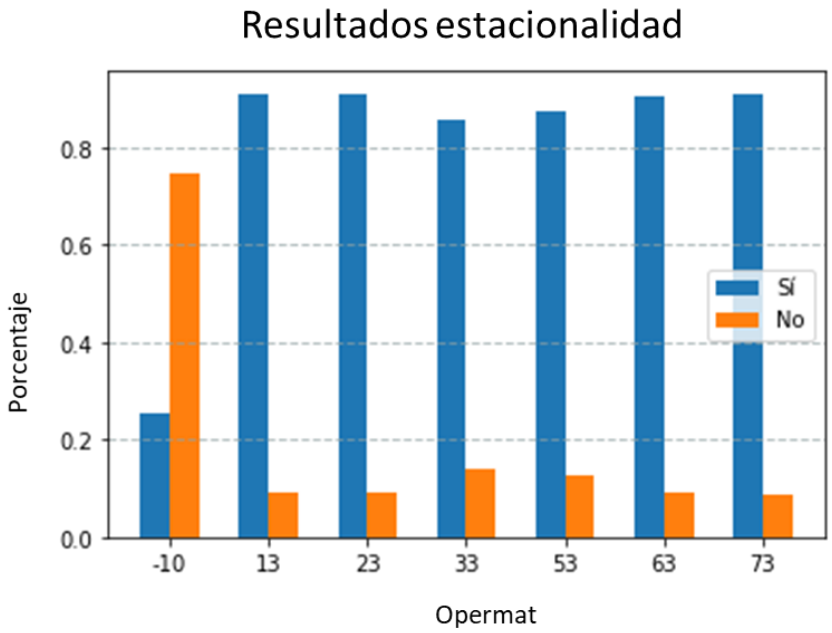


Figura 5.7: Representación del resultado de la prueba de estacionalidad. Sí indica que son estacionarias las series.

[2]Consúltese el Capítulo 20 de [22]
[3]Véase la sección 20.3 de [22]

5.3. Arquitectura del Modelo

En esta sección se describen los detalles sobre la elección y construcción de la técnica de clasificación.

Como resultado del estudio inicial (*Sección 5.1*) se tiene que las operaciones realizadas, siendo del mismo tipo y sobre el mismo material, pueden ser muy heterogéneas en cuanto a la duración se refiere. Esto se debe a que una operación depende de ciertas características del material (grosor, superficie, forma, etc.) así como del plano de la pieza.

Con objeto de minimizar el efecto de las características anteriores (detectados como deformaciones en el tiempo) se recurre a la distancia DTW para la comparación de series. Sin embargo, la DTW presenta dos problemas asociados:

- El coste computacional crece polinómicamente con orden 5 con respecto al número de observaciones.
- La DTW puede producir deformaciones temporales excesivas en la comparación (una duración excesiva también puede deberse a una anomalía).

Para minimizar estos problemas, se recurre a un enfoque de clasificación híbrido (véase *Sección 4.6*). Por tanto, el modelo que se construye realiza una doble clasificación:

Paso 1. Se clasifica el conjunto total de las actividades de un tipo concreto, representada cada una por la correspondiente serie temporal, denotado por $\mathcal{D}$, atendiendo exclusivamente a su duración mediante el algoritmo de k-medias tradicional. Así, se obtiene una primera descomposición $\mathcal{D} = \mathcal{D}_1 \sqcup \cdots \sqcup \mathcal{D}_p$.

Paso 2. Las series temporales de cada uno de los subconjuntos $\mathcal{D}_1, \ldots, \mathcal{D}_p$ anteriores son clasificadas mediante el algoritmo de k-medias bajo la DTW. En consecuencia, se descompone $\mathcal{D}_i = K_{i,1} \sqcup \cdots \sqcup K_{i,s}$, luego,

$$\mathcal{D} = \bigsqcup_{i=1}^{p} \bigsqcup_{j=1}^{s} K_{i,j}$$

A los subconjuntos $K_{i,j}$ les denominaremos **clústeres finales**.

La técnica híbrida de clasificación descrita se resume gráficamente en la *Figura 5.8*.

5.3.1. Los parámetros del modelo

Por tanto, el modelo está caracterizado por dos parámetros, el número de clústeres en los que se divide atendiendo a la duración, denotado por p, y el número de clústeres en los que se divide analizando las series temporales bajo la DTW, denotado por s. Estos parámetros son determinados experimentalmente en la *Sección 9.1*, y forman lo que se denomina la **configuración para la clasificación**.

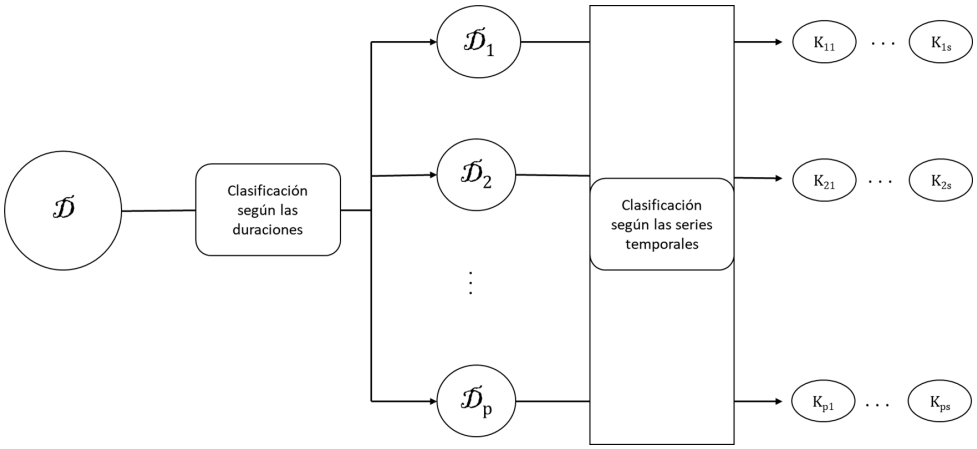


Figura 5.8: Esquema del modelo de clasificación híbrido planteado

Por otro lado, dado que el objetivo principal es detectar las actividades anómalas comparando sobre un conjunto de realizaciones sin valoración, se requiere determinar si un clúster es usual o anómalo. En el caso de estudio de esta memoria, se dice que un clúster **es usual** si el porcentaje de realizaciones asignadas a dicho clúster respecto del total es mayor que un cierto valor α. En caso contrario, el clúster **es anómalo**.

Como consecuencia, surge un tercer parámetro a determinar, denominado **tolerancia** y denotado por α, encargado de discriminar entre usual y anómalo. Los tres parámetros anteriores forman lo que se denomina la *configuración del modelo*, y el valor de tolerancia también es determinado experimentalmente en la *Sección 9.2.1*.

5.3.2. El proceso de evaluación según un modelo

Al igual que para la construcción del modelo, el proceso de evaluación de una serie temporal de muestra consta de dos pasos:

Paso 1: Asignación de un clúster por la duración. Se considera la duración de la serie y se asigna un clúster según el centroide de mínima distancia para la clasificación por k-medias (tradicional).

Paso 2: Asignación de clúster final. Se toma la división por k-medias bajo la DTW para el subconjunto asignado según la duración en el paso anterior, y se asigna un clúster según la mínima distancia (de la DTW) a los centroides.

Finalmente, una operación es considerada usual o anómala cuando así lo sea su clúster final asignado.

Cada tipo de operación sobre cada tipo de material es esencialmente diferente al resto. Como consecuencia, se requiere construir su propio Modelo de Clasificación para cada actividad estudiada: Planteado, Contorneado, Fresado y Taladrado sobre el acero (*opermat* de 23, 33, 53, 63 respectivamente) y arranques de la máquina (identificados por *opermat* de -10).

Capítulo 6

Arquitectura del proyecto

En este capítulo se describe la arquitectura seguida en el desarrollo del sistema. Se va a diseñar una aplicación modular, donde cada módulo cumple con una funcionalidad concreta, satisfaciendo unos objetivos determinados, y, para su cumplimiento, intercambia información con el resto de los módulos.

6.1. Una aplicación de dos sistemas

El sistema planteado se divide en dos grandes componentes, construyendo así un sistema básico capaz de procesar las actividades, evaluarlas y almacenar de forma estructurada los resultados, y un conjunto de funcionalidades extras tales como las interfaces de usuario o el sistema experimental.

- El **sistema principal** o ***script* de clasificación**, que comprende las funcionalidades principales: lectura de los datos, procesado de la información, generación del modelo, diagnosis de las actividades realizadas según el modelo generado para detección de anomalías y almacenado de las anomalías en la base de datos para su lectura por el sistema externo de representación.
- El **sistema secundario** comprende las funcionalidades necesarias para la realización del estudio experimental y análisis de las operaciones, así como una interfaz para interactuar con el sistema principal y visualizar los resultados de la clasificación en vivo.

Esta división viene dada como un requisito del proyecto fijado por la Fundación Cidaut, pues dentro del proyecto **IDIGIT4L** se trabajará con un *script* capaz de ejecutarse de manera independiente en un sistema específico que realiza las mediciones, y requiere de procesar la información para detectar las anomalías y que sean representadas mediante el software *Grafana* (dado por el **sistema principal**). Por tanto, para complementar este componente no se requiere ni de una interfaz ni de las funcionalidades que realizan el estudio experimental.

Con esta división se consigue que el sistema principal sea completamente funcional e independiente del sistema secundario, que lo dota de aquellas funcionalidades necesarias para completar las características del *script* principal (estudio experimental de los parámetros) e interactuar con el usuario (interfaces de usuario).

6.2. Características principales

En esta sección se describen las capacidades principales que ofrece el Sistema de Detección de Anomalías. Dichas capacidades se recogen en la *Tabla 6.1*.

Id	Característica
CAR-01	**Creación del modelo:** el sistema tendrá la capacidad de crear un modelo de clasificación como el descrito en la *Sección 5.3* según una serie de parámetros predefinidos, utilizando la información almacenada en la base de datos.
CAR-02	**Validación de las actividades en vivo**: el sistema tendrá la capacidad de leer la información recogida en el momento presente de la base de datos, identificar el estado de la máquina, evaluar las actividades realizadas según el modelo y notificar las incidencias detectadas.
CAR-03	**Estudio experimental:** el sistema proveerá de todas las herramientas necesarias para la realización de un estudio que permita determinar de forma experimental la *configuración del modelo*.

Tabla 6.1: Características del Sistema de Detección de Anomalías

6.3. Módulos de cada sistema

En esta sección se describe cada uno de los módulos que componen el sistema, discriminando por su pertenencia al sistema principal o al sistema secundario.

El sistema principal está compuesto por los siguientes módulos:

- **Módulo de conexión e intercambio con la base de datos**, encargado de conectar con la base de datos, leer la información recogida por los sensores y almacenar las anomalías detectadas en las tablas correspondientes.

- **Módulo de procesamiento y estructurado de la información**, encargado de procesar la información extraída de la base de datos o leída en vivo (cargada en ambos casos como una tabla `pandas.DataFrame`) para su estructuración según las clases correspondientes.

- **Módulo de clasificación**, que implementa el modelo de clasificación descrito en la *Sección 5.3*, con las funcionalidades para entrenarlo con un conjunto de actividades, valorar el propio modelo de clasificación y evaluar una actividad según su modelo.

- **Módulo de evaluación de actividades en vivo**, encargado de tomar cada segundo la última medición de los sensores, identificar el estado de la máquina (apagada, en arranque, en operación o encendida sin actividad-ociosa) y evaluar una actividad cuando esta concluye según el modelo para detectar las anomalías correspondientes. Además, guarda cierta información sobre el estado de la máquina y la última operación en un archivo `json` para su uso en la interfaz.

- **Módulo de persistencia**, encargado de almacenar las actividades procesadas y los modelos de forma persistente en archivos binarios `.pickle`, así como de recuperar la información almacenada.

Y el sistema secundario se compone de los siguientes módulos:

- **Módulo del estudio experimental**, que comprende las funcionalidades necesarias para realizar el estudio experimental para determinar la configuración óptima para la clasificación (parámetros p, s en la *Sección 5.3.1*) y para el valor de tolerancia (α en la *Sección 9.2.1*).

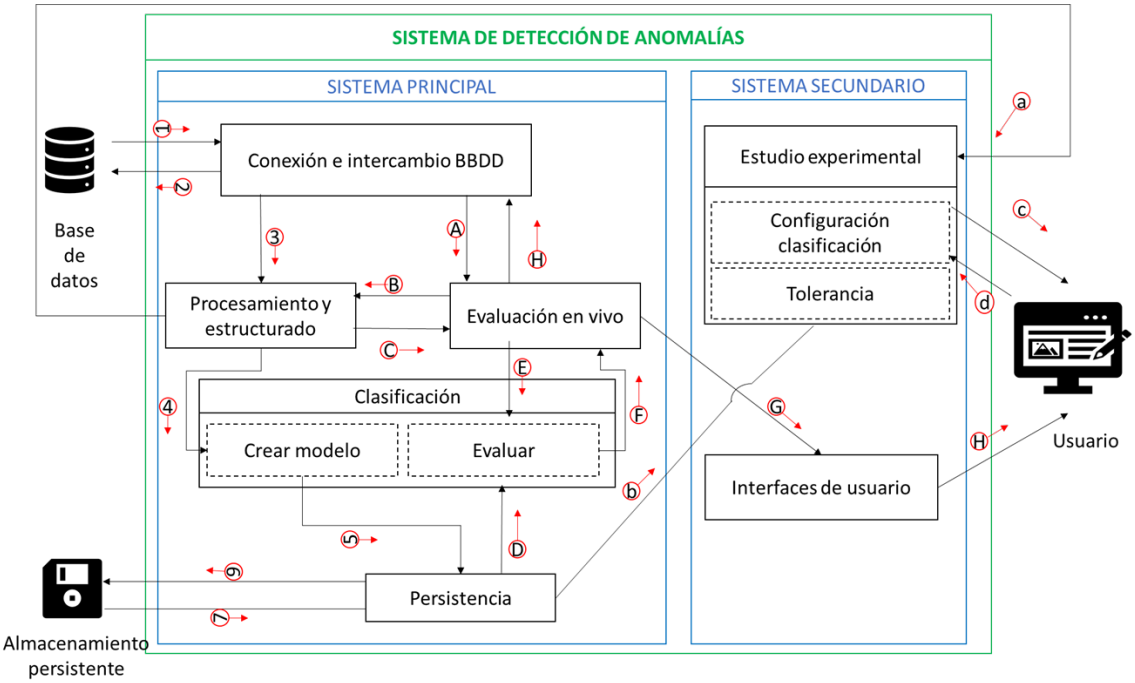
- **Módulo de interfaz de usuario**, que comprende todas las interfaces del sistema desarrollado: interfaces para el estudio experimental, interfaz para visualizar los resultados del análisis de las actividades en vivo

El siguiente diagrama de estructura[1] (*Figura 6.1*) describe los diferentes módulos del proyecto y sus interacciones mediante el intercambio de información.

La comunicación entre los módulos es necesaria para cumplir con las características recogidas en la *Tabla 6.1*. Los siguientes diagramas de secuencia[2] (*Figuras 6.2, 6.3 y 6.4*) ilustran el orden en que se produce dicha comunicación, utilizando la notación de la *Figura 6.1*.

[1] Véase [23] diagramas de estructura (*Structure Diagrams*), sección B.2, y estructuras de clasificación en la sección 11

[2] Véase [23], sección 17.8



1) Consulta sobre la base de datos
2) Incidencias detectadas
3) Información de la consulta tabulada
4) Conjunto de actividades estructuradas
5) Modelo de clasificación
6) Archivo pickle
7) Archivo pickle

A) Información de la consulta tabulada
B) Información tabulada de una actividad
C) Actividad estructurada
D) Modelo de clasificación
E) Actividad representada como clase
F) Resultado de la evaluación
G) JSON con el resultado de la evaluación
H) Actividad anómala tabulada

a) Conjunto de actividades estructuradas
b) Solicitud de información e información relevante para el usuario
c) Información introducida por el usuario

Figura 6.1: Diagrama de estructura del Sistema de Detección de Anomalías

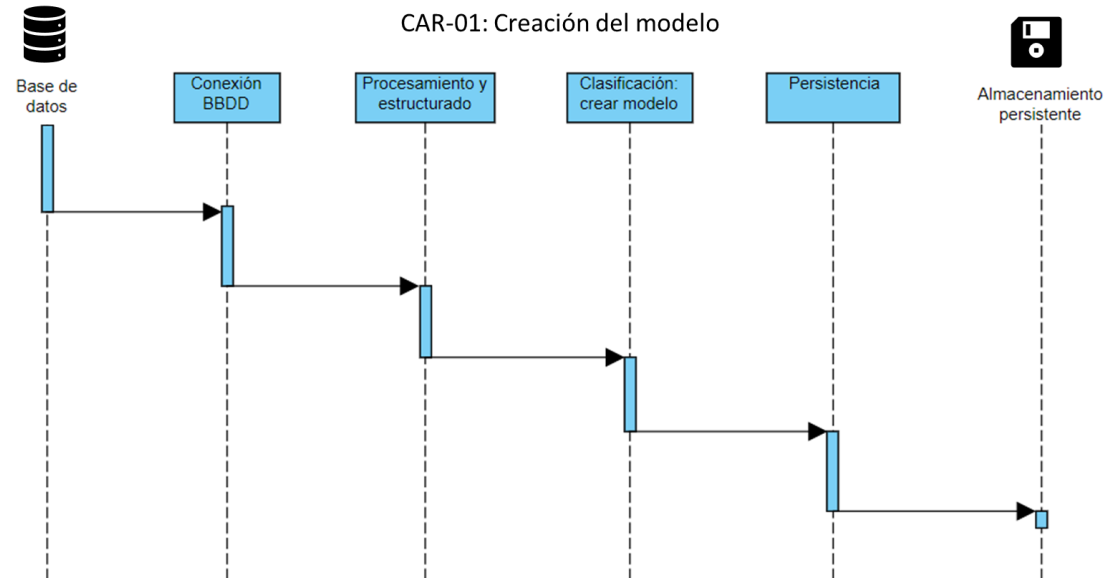


Figura 6.2: Diagrama de secuencia para la característica CAR-01, donde los mensajes enviados mantienen la notación utilizada en la *Figura 6.1*.

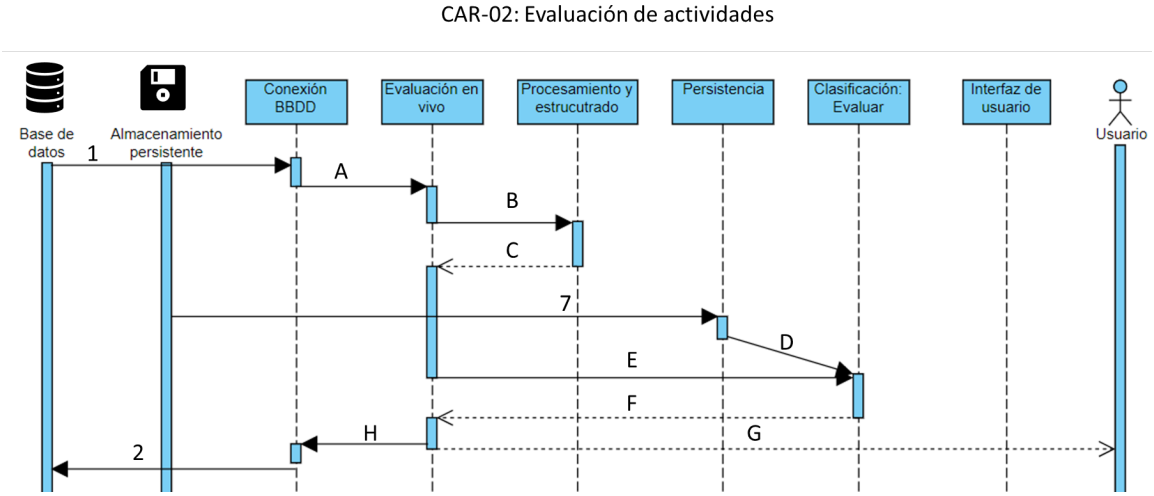


Figura 6.3: Diagrama de secuencia para la característica CAR-02, donde los mensajes enviados mantienen la notación utilizada en la *Figura 6.1*.

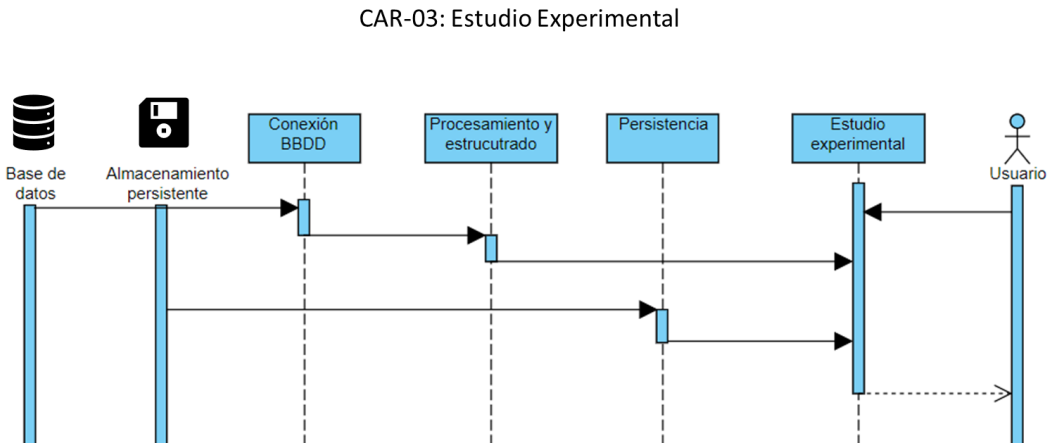


Figura 6.4: Diagrama de secuencia para la característica CAR-03, donde los mensajes enviados mantienen la notación utilizada en la *Figura 6.1*.

Capítulo 7

Análisis y diseño

En este capítulo se abordan los detalles más relevantes sobre el análisis y diseño llevados a cabo para el desarrollo del Sistema de Detección de Anomalías (SDA). Se va a construir un sistema modular, donde cada módulo es independiente del resto y cumple ciertas funcionalidades concretas, específicas y exclusivas. Para el diseño y la implementación del modelo y de la estructuración de las actividades se va a optar por un diseño orientado a objetos.

Cabe destacar que, el presente trabajo, no es un diseño de una aplicación de software al uso, sino que contempla el estudio para la elección de un modelo de clasificación y su aplicación sobre un entorno industrial cambiante dentro de un proyecto mayor con ciertos requisitos externos sobre el cual se debe integrar.

7.1. Requisitos de Información

En esta sección se recoge la información relativa a los tipos de datos con los que trabaja el Sistema de Detección de Anomalías (SDA).

Cabe destacar que, ambos sistemas (principal y secundario) son una componente dentro de un proyecto mayor. Por ello, la recogida y almacenamiento de la información (estructura de la base de datos) y la visualización de los resultados son realizadas por agentes externos.

Los requisitos de información están descritos en la *Tabla 7.1*. De estos requisitos, los identificado por **RI-01**, **RI-02** y **RI-03** hacen referencia a la información recogida en la base de datos y sus restricciones. El requisito **RI-04** incluye toda la información que es necesaria para la construcción de los modelos y evaluación de actividades según dicho modelo. Los requisitos **RI-05** y **RI-06** contemplan como el programa va a estructurar la información extraída de la base de datos y la información de los modelos respectivamente. Y, el requisito **RI-07**, representa el estado de la máquina fresadora.

Id	Requisito de información
RI-01	Una **Medición** está compuesta por toda la información registrada por los sensores en un instante dado, por la información introducida por el usuario (operación, material, herramienta y orden de fabricación) y por el instante concreto en el que se efectúa. Se registra en la tabla `CNC_Funcionamiento`
RI-02	Una **Realización** está compuesta por la fecha de comienzo, la fecha de finalización, el tipo de operación y herramienta y la orden de fabricación de cada operación. Se registra en la tabla `TiempoOperaciones`
RI-03	Una **Incidencia** está compuesta por el instante de realización, el tipo de operación, material y herramienta y la orden de fabricación. Se registra en la tabla `CNC_Incidencias`
RI-04	Una **Configuración** hace referencia al conjunto de parámetros predefinidos necesarios para la construcción del modelo y la evaluación de las actividades. Se registran en un archivo de Python denominado `configuracion.py`.
RI-05	Una **Actividad** es el resultado de agrupar y estructurar todas las mediciones asociadas a una misma operación o un mismo arranque.
RI-06	Un **Modelo de Clasificación** es una estructura de clasificación por k-medias como la descrita en la *Sección 5.3*, y entrenada a partir de un conjunto de actividades de un único tipo de *opermat*.
RI-07	El **Balance de Situación** indica cómo se encuentra la máquina fresadora CNC en un instante dado (estado, evaluación de la última operación, etc.)

Tabla 7.1: Requisitos de Información del Sistema de Detección de Anomalías

El siguiente diagrama conceptual (*Figura 7.1*) modela los requisitos de información y sus relaciones. Estas relaciones entre los requisitos de información están descritas en la *Tabla 7.2*.

Id	Requisito de información
Rel-01	Una Realización de una operación esta **asociada** con una Actividad que la representa, y una Actividad puede representar a lo sumo a una única Realización.
Rel-02	Una Realización esta **compuesta** por varias Mediciones y una Medición compone a lo sumo una única Realización.
Rel-03	Una Actividad esta **compuesta** por varias Mediciones y una Medición compone a lo sumo una única Actividad.
Rel-04	Una Incidencia esta **asociada** a una única Mediciones y una Medición puede estar asociada a lo sumo una única Incidencia.
Rel-05	Una Actividad **interactúa** (entrena o es evaluada) por un único Modelo (asociado por tipo de *opermat*) y un Modelo interactúa con varias Actividades.
Rel-06	Una configuración **determina** algún Modelo de clasificación y un Modelo tiene una y solo una Configuración.

Tabla 7.2: Relaciones entre los requisitos de información

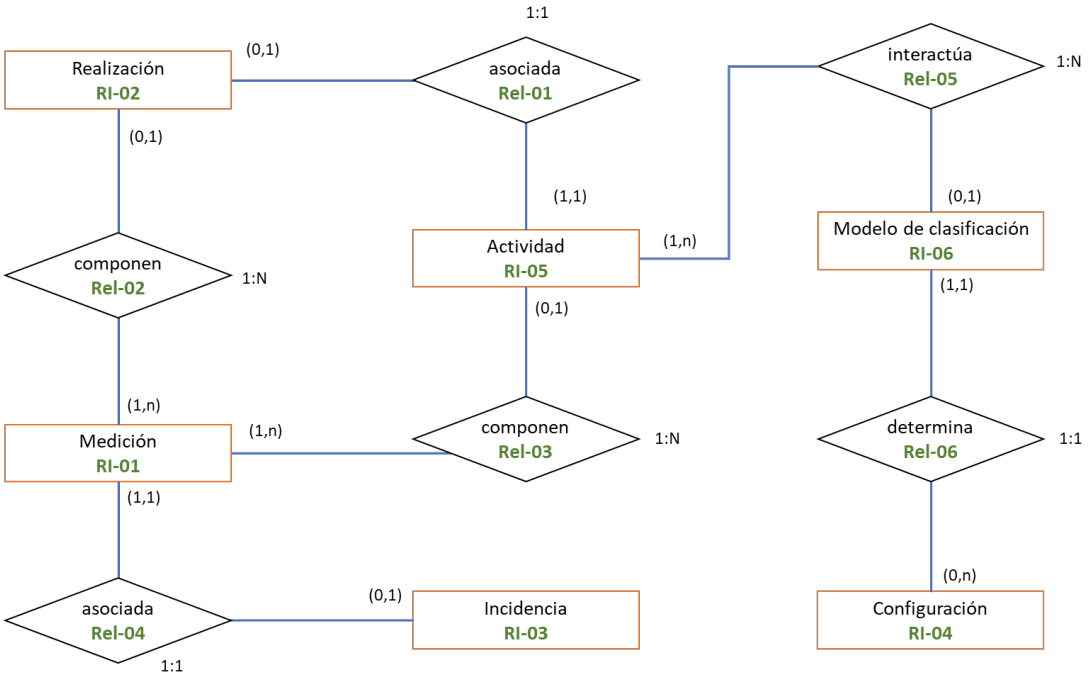


Figura 7.1: Diagrama Entidad-Relación para modelar los requisitos de información y sus relaciones

Los requisitos de información **RI-01** y **RI-02** son descritos en las *Tablas 3.6 y 3.7*, en la *Sección 3.3*, donde se describió la estructura de las tablas `CNC_Funcionamiento` y `tiempoOperaciones`, que estructuran dichos requisitos. El requisito **RI-03** es descrito en la próxima sección (7.1.1), donde se presenta la nueva tabla `CNC_Incidencias` (*Tabla 7.3*) creada para almacenar las incidencias detectadas y que sean representadas por el software *Grafana*.

Los requisitos **RI-04**, **RI-05** y **RI-06** son descritos en las *Tablas 7.4, 7.5 y 7.6*. En la *Sección 7.3* se encuentran los detalles de cómo se definen y estructuran dichos requisitos de información en el Sistema de Detección de Anomalías, puesto que no forman parte de la base de datos propia.

7.1.1. La base de datos completa

En esta sección se establecen de forma detallada y precisa los requisitos de información del sistema. En consecuencia, esta sección está dedicada a la base de datos (que es externa al sistema de clasificación) con objeto de recoger y conocer cómo se estructura la información en bruto, y que posteriormente será procesada. Los requisitos de informa

El siguiente modelo conceptual (*Figura 7.2*) describe los diferentes tipos de entidades que modelizan la información leída por los sensores y almacenada en la base de datos, así como las relaciones que la modelan. En la *Sección 3.3* se recoge una imagen del estado de la base de datos

antes de comenzar el proyecto y se describen los requisitos de información **RI-01** y **RI-02**. En esta sección se desarrolla el requisito **RI-03**, completando la base de datos con la que interactúa el Sistema de Detección de Anomalías.

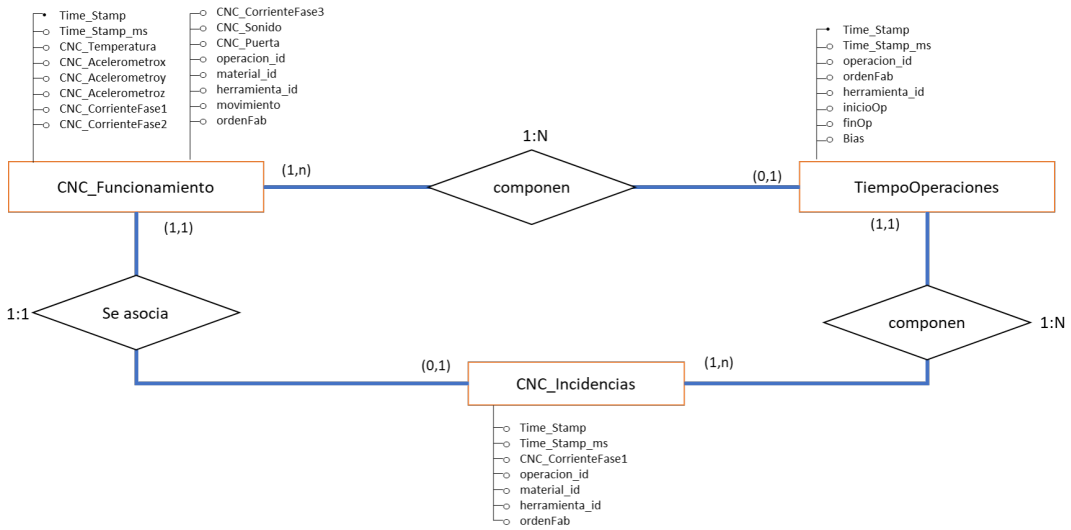


Figura 7.2: Diagrama entidad-relación correspondiente a la base de datos completa

Este nuevo requisito de información **RI-03** tiene asociados los siguientes atributos, recogidos en la *Tabla 7.3* y que se corresponde con la tabla `CNC_Incidencias` de la base de datos.

La estructura de esta tabla no es arbitraria, y contiene la información necesaria para que el software *Grafana*, encargado de la visualización de los resultados dentro del proyecto general, pueda representar las incidencias en las situaciones deseadas según el sensor de corriente de fase 1.

En cuanto a las relaciones con las otras dos tablas de uso en el sistema, cada entrada de `CNC_Incidencias` está asociada con una y solo una fila de `CNC_Funcionamiento` y cada entrada de `CNC_Funcionamiento` se asocia a lo sumo una fila de `CNC_Incidencias`. Esta relación viene a indicar si una medición de `CNC_Funcionamiento` es considerada anómala o no, dependiendo respectivamente de si se asocia o no a una entrada en `CNC_Incidencias`. Análogamente, y dado que en el sistema planteado no se analizan las mediciones por separado, sino que se analiza el conjunto de mediciones de toda la realización de la actividad, entonces cada operación anómala (una entrada en `TiempoOperaciones`) está compuesta por el conjunto de entradas de `CNC_Incidencias`, y una operación usual no tiene ninguna entrada en la tabla de incidencias.

Campo BBDD	Introduce información	Tipo de dato y rango de valores	Notas
`Time_Stamp`	*Script* de clasificación	Cadena de tiempo con formato "año-mes-día hora:min:seg"	Clave primaria y ajena respecto de la tabla `CNC_FUNCIONAMIENTO`
`Time_Stamp_ms`	*Script* de clasificación	Entero positivo	Valor de control. Actualmente sin uso.
`CNC_Corriente_Fase1`	*Script* de clasificación	Real	Medición realizada por el sensor de corriente de fase 1 para el instante dato en `Time_Stamp`.
`operacion_id`	*Script* de clasificación	Entero del 1 al 7	Tipo de operación realizada.
`material_id`	*Script* de clasificación	Entero del 1 al 5	Tipo de material trabajado.
`herramienta_id`	*Script* de clasificación	Entero del 1 al 31	Tipo de herramienta utilizada.
`orden_fab`	*Script* de clasificación	Cadena de caracteres	Orden de fabricación asociada con la acción realizada

Tabla 7.3: Campos e información de la tabla `CNC_Incidencias`

7.1.2. Descripción de los requisitos de información

En esta sección se desarrollan los requisitos funcionales **RI-04**, **RI-05** y **RI-06** que no forman parte de la base de datos y aportan información sobre el sistema.

El requisito **RI-04**, **Configuración**, hace referencia a los parámetros necesarios para el funcionamiento del Sistema de Detección de Anomalías, incluyendo la configuración del modelo de clasificación y el valor de tolerancia.

El requisito **RI-05**, **Actividad**, hace referencia a la información extraída de la base de datos que ha sido estructurada y procesada. Así, se eliminan los datos que carecen de interés y se mantiene aquella información útil de una forma ordenada, regularizada y organizada. Es decir, este requisito de información hace referencia a la forma en que el sistema da uso a la información de la base de datos.

El requisito **RI-06**, **Modelo de Clasificación**, se refiere al propio modelo de clasificación (*Sección 5.3*), y muestra la primera aproximación a su diseño y estructura.

Nombre	Tipo de dato	Descripción
`operación_n_clusters_dur` (**RI-04.1**)	Entero positivo	Número de clústeres en los que se clasifica por k-medias atendiendo a la duración de las series temporales (parámetro p de la *Sección 5.3.1*) para las operaciones.
`operación_n_clusters_ts` (**RI-04.2**)	Entero positivo	Número de clústeres en los que se divide, cada conjunto separado por la duración de las series, según el algoritmo de k-medias bajo la DTW (parámetro s de la *Sección 5.3.1*) para las operaciones.
`operacion_dur_min` (**RI-04.3**)	Entero positivo	Duración mínima (en segundos) exigida para que una serie temporal, que representa una operación, sea considerada en el entrenamiento del modelo.
`operacion_dur_max` (**RI-04.4**)	Entero positivo	Duración máxima (en segundos) permitida para que una serie temporal, que representa una operación, sea considerada en el entrenamiento del modelo.
`arranque_n_clusters_dur` (**RI-04.5**)	Entero positivo	Número de clústeres en los que se clasifica por k-medias atendiendo a la duración de las series temporales (parámetro p de la *Sección 5.3.1*) para los arranques.
`arranque_n_clusters_ts` (**RI-04.6**)	Entero positivo	Número de clústeres en los que se divide, cada conjunto separado por la duración de las series, según el algoritmo de k-medias bajo la DTW (parámetro s de la *Sección 5.3.1*) para los arranques.
`arranque_dur_min` (**RI-04.7**)	Entero positivo	Duración mínima (en segundos) exigida para que una serie temporal, que representa un arranque, sea considerada en el entrenamiento del modelo.
`arranque_dur_max` (**RI-04.8**)	Entero positivo	Duración máxima (en segundos) permitida para que una serie temporal, que representa un arranque, sea considerada en el entrenamiento del modelo.
`OPERMATS_MODELO` (**RI-04.9**)	Lista	Conjunto de *opermats* para los cuales se va a crear un modelo de clasificación propio.
`tolerancia_operaciones` (**RI-04.10**)	Real positivo entre 0 y 1	Porcentaje de operaciones (respecto del total con el que se entrena) mínimo que debe haber en un clúster para ser considerado usual.
`tolerancia_operaciones` (**RI-04.11**)	Real positivo entre 0 y 1	Porcentaje de arranques (respecto del total con el que se entrena) mínimo que debe haber en un clúster para ser considerado usual.
`fecha_ini` (**RI-04.12**)	Cadena de caracteres	Fecha inicial a partir de la cual se toman los datos para la construcción del modelo.

Tabla 7.4: Parámetros de configuración **RI-04**

Nombre	Tipo de dato	Descripción
`corriente1` (**RI-05.1**)	Serie temporal real	Representa las mediciones del sensor de fase de corriente 1 durante la totalidad de la actividad.
`corriente2` (**RI-05.2**)	Serie temporal real	Representa las mediciones del sensor de fase de corriente 2 durante la totalidad de la actividad.
`corriente3` (**RI-05.3**)	Serie temporal real	Representa las mediciones del sensor de fase de corriente 3 durante la totalidad de la actividad.
`acelerómetrox` (**RI-05.4**)	Serie temporal real	Representa las mediciones del acelerómetro-eje-X durante la totalidad de la actividad.
`acelerómetroy` (**RI-05.5**)	Serie temporal real	Representa las mediciones del acelerómetro-eje-Y durante la totalidad de la actividad.
`acelerómetroz` (**RI-05.6**)	Serie temporal real	Representa las mediciones del acelerómetro-eje-Z durante la totalidad de la actividad.
`operación` (**RI-05.7**)	Entero positivo	Código del tipo de operación realizada (–1 para los arranques de la máquina).
`material` (**RI-05.8**)	Entero positivo	Código del tipo de material trabajado.
`herramienta` (**RI-05.9**)	Entero positivo	Código del tipo de herramienta utilizada.
`opermat` (**RI-05.9**)	Entero positivo	Código del tipo de *opermat* asignado, dado como 10 veces el código de `operación` más el código de `materia`.
`Fecha` (**RI-05.10**)	Cadena de caracteres	Fecha y hora exacta de comienzo de la actividad.
`orden_fab` (**RI-05.11**)	Cadena de caracteres	Identificador interno de la Fundación Cidaut que determina el plano que se utiliza para la fabricación de la pieza.

Tabla 7.5: Parámetros de configuración **RI-05**

Nombre	Tipo de dato	Descripción
`opermat` (**RI-05.1**)	Entero positivo	Tipo de *opermat* de las series temproales con las que se entrenó el modelo de clasificación híbrido
`n_clústeres_dur` (**RI-05.2**)	Entero positivo	Número de clústeres usados para dividir las series temporales por sus duraciones
`n_clústeres_ts` (**RI-05.3**)	Entero positivo	Número de clústeres usados para clasificar cada subconjunto obtenido al dividir por la duración por el algoritmo de k-medias bajo la DTW.
`clasificación` (**RI-05.4**)	Indeterminado	Representa el resultado de clasificar los datos según el modelo híbrido (centroides para la duración y para la DTW, elementos de cada conjunto, etc.). Se desarrolla con detalle en la clase `Kmedias_tiempo` de la *Sección 7.3*.

Tabla 7.6: Parámetros de configuración **RI-06**

El último requisito **RI-07**, el **Balance de Estado**, hace referencia a la situación en que se encuentra la máquina fresadora en el instante actual.

El siguiente diagrama de estados[1] (*Figura 7.3*) ilustra los cambios del requisito, **Estado RI-07.1** en las diferentes situaciones que se producen sobre la máquina fresadora.

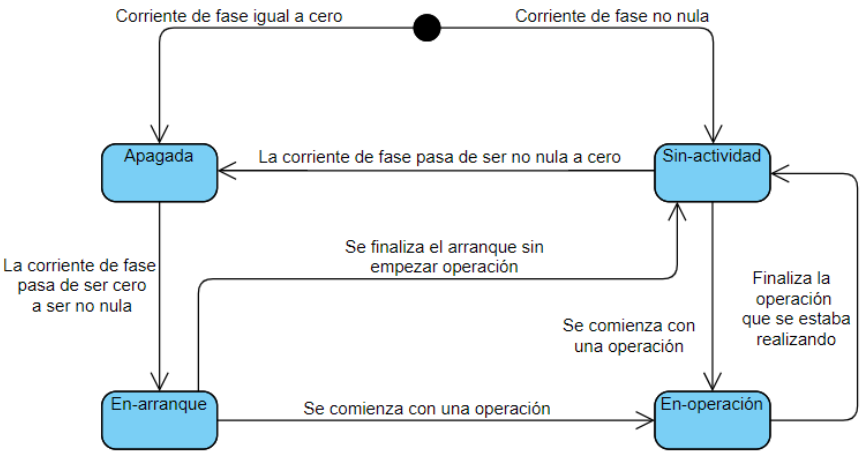


Figura 7.3: Diagrama de estados para el requisito de información **RI-05**

[1]Véase [23] secciones 14.2 (*Composite State*) y B.7

Nombre	Tipo de dato	Descripción
`estado` (**RI-07.1**)	Lista de opciones	Determina la situación en la máquina dada por los siguientes valores. ▪ *Apagada*: corriente de fase igual a cero. ▪ *En-operación*: realizando algún tipo de operación concreto. ▪ *En-arranque*: iniciándose y preparándose antes de poder ser usada. ▪ *Libre o sin-actividad*: sin operación ni arranque, situación entre operaciones.
`operación_actual` (**RI-07.2**)	Entero positivo	Código identificativo del tipo de operación en curso.
`material_actual` (**RI-07.3**)	Entero positivo	Código identificativo del tipo de material en uso.
`operación_anterior` (**RI-07.4**)	Entero positivo	Código identificativo del último tipo de operación realizada.
`operación_anterior` (**RI-07.5**)	Entero positivo	Código identificativo de tipo de operación de la anterior actividad.
`material_anterior` (**RI-07.6**)	Entero positivo	Código identificativo del último material utilizado en la anterior actividad.
`resultado_anterior` (**RI-07.6**)	Lista de opciones	Resultado de la evaluación de la última actividad según el modelo: usual o anómala.

Tabla 7.7: Parámetros de configuración **RI-06**

7.2. Funcionalidades del sistema: Requisitos de Usuario

En esta sección se describen, con un alto nivel de abstracción, las funcionalidades que ofrece el Sistema de Detección de Anomalías a los diferentes usuarios que interactúan con él y como las ofrece.

7.2.1. Actores del sistema

En esta sección se describirán todos los actores (agentes externos tales como personas o sistemas software) que interactúan con el Sistema de Detección de Anomalías, tanto principales (ejecutan alguna acción con el sistema) como secundarios (proveen o reciben información del sistema).

Del sistema principal, cabe destacar que se plantea como un sistema autónomo, de continuo

funcionamiento y donde la interacción con el usuario es mínima, siendo necesaria exclusivamente para fijar la configuración creación del primer modelo o para ejecutar por primera vez el *script de clasificación* (acciones asociadas al entorno software sobre el que se ejecuta).

Por tanto, el sistema principal presenta un único actor, denominado **Software Huésped**, que se corresponde al entorno sobre el cual se ejecuta dicho *script*. Por otro lado, el sistema completo presenta un único actor denominado **Usuario**, y que se corresponde con la persona que interactúa con las interfaces.

Además, existe un actor secundario a través del que se obtiene la información recogida por los sensores y que representa las tablas `CNC_Funcionamiento` y `tiempoOperaciones` de la base de datos. La *Tabla 7.8* recoge los actores del sistema.

Id	Nombre	Tipo de actor	Descripción
ACT-01	Software Huésped	Principal	Software sobre el que se ejecuta el sistema principal, encargado de lanzar las funcionalidades de creación del modelo y evaluación en vivo.
ACT-02	Usuario	Principal	Rol general que describe a toda persona que interactúan con el SDA mediante el subsistema secundario.
ACT-03	Base de datos	Secundario	Tablas en las cuales los sensores registran la información. Proveedor del SDA. de la información necesaria.

Tabla 7.8: Actores del Sistema de Detección de Anomalías

7.2.2. Requisitos de Usuario

Los requisitos de usuario declaran los servicios y funcionalidades que el sistema ofrece y las restricciones bajo las que opera.

Para el desarrollo del sistema se va a optar por un modelo evolutivo e incremental, dividido en varias fases. En cada una de estas fases se trabaja con uno o dos requisitos de usuario, desarrollando e implementando las funcionalidades necesarias. En la próxima sección (7.2.3) se describen en detalle estos requisitos de usuario como casos de uso, y en la *Capítulo 8* se describen las diferentes fases del proyecto para cada caso de uso, junto a sus requisitos funcionales y otras consideraciones de diseño.

En la *Tabla 7.9* se listan los requisitos de usuario que debe satisfacer el sistema y los actores principales (que solicitan interactuar con el Sistema de Detección de Anomalías) de dichos re-

quisitos, y en la *Figura 7.4* se ilustran mediante un diagrama de casos de uso[2] los requisitos de usuario y sus interacciones con los actores del sistema y con otros requisitos.

Id	Actor	Nombre
RU-01	Software Huésped	Generar el modelo de clasificación
RU-02	Software Huésped	Evaluar el comportamiento de la máquina fresadora en vivo
RU-03	Usuario	Generar un modelo con los parámetros modificados por el usuario
RU-04	Usuario	Visualizar el estado actual de la máquina
RU-05	Usuario	Realizar el estudio experimental de la *configuración para la clasificación*
RU-06	Usuario	Realizar el estudio experimental del valor de *tolerancia* adecuado

Tabla 7.9: Actores del Sistema de Detección de Anomalías

7.2.3. Casos de uso

En esta sección se describen los requisitos de usuario recogidos en la *Tabla 7.9* como casos de uso, recogidos en las *Tablas 7.10, 7.11,7.12,7.13,7.14 y 7.15,*, y representados mediante diagramas de actividad[3] en las *Figuras 7.5, 7.6, 7.7, 7.8, 7.9 y 7.10.*

[2]Véase [23] sección 18.1, *Use Cases*
[3]Consúltese [23] secciones 15.2, 15.3 y 15.4 *Activities*

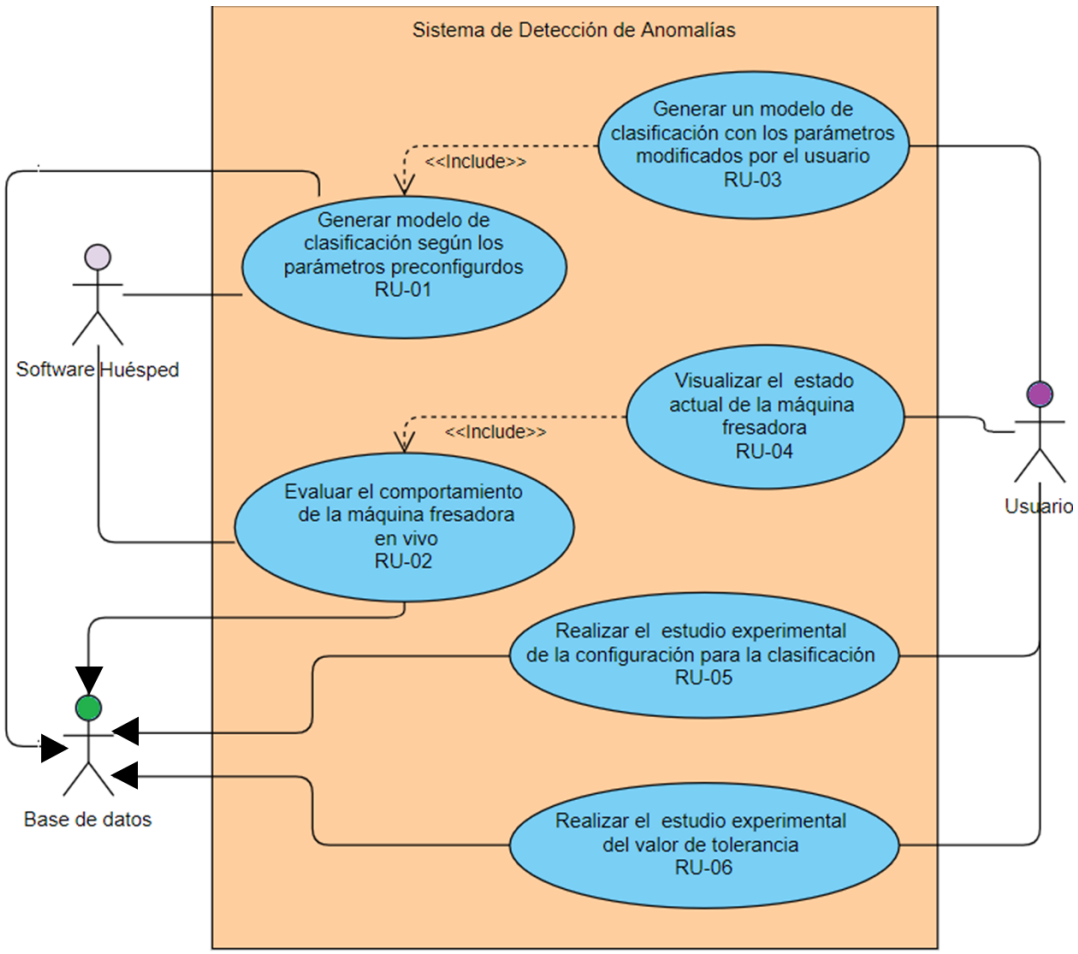


Figura 7.4: Diagrama de casos de uso del Sistema de Detección de Anomalías

RU-01	**Generar el modelo de clasificación según parámetros preconfigurados**
Actor(es)	Software Huésped (**ACT-01**) y Base de Datos (**ACT-03**)
Descripción	El SDA deberá comportarse como indica el siguiente caso de uso cuando el *Software Huésped* solicite crear un nuevo modelo de configuración.
Precondición	Existen unos parámetros predefinidos requeridos en el paso segundo de la secuencia normal
Flujo normal	1. El Software Huésped solicita generar un nuevo modelo de clasificación. 2. El sistema lee los parámetros preconfigurados (**RI-04**): fecha inicial (**RI-04.12**), numero de clústeres para la duración (**RI-04.1** y **RI-04.5**), número de clústeres para las series temporales (**RI-04.2** y **RI-04.6**), *opermat* de las actividades de interés a generar el modelo (**RI-04.9**)y duraciones máximas y mínimas de las operaciones (**RI-04.3**, **RI-04.4**, **RI-04.7** y **RI-04.8**). 3. El sistema solicita a la *Base de Datos* toda la información disponible a partir de una fecha inicial dada, **RI-01** y **RI-02** (**módulo de conexión e intercambio BBDD**). 4. El sistema procesa y estructura la información obtenida de la base de datos, **RI-04** (**módulo de procesamiento y estructurado de la información**) . 5. El sistema separa las actividades de interés por su *opermat*, depura por las duraciones y para cada tipo de actividad genera su modelo (**RI-06**) correspondiente (**módulo de clasificación**). 6. El sistema guarda de forma persistente los modelos como archivos `pickle` (**módulo de persistencia**). 7. El caso de uso finaliza satisfactoriamente
Flujos alternos	2. Si se produce un error con la base de datos el sistema lo notifica y finaliza el caso de uso con error.
Postcondición	Generado el modelo de clasificación para los arranques (denominado `modelo_arranque`) y para las operaciones (`modelo_operaciones`).

Tabla 7.10: Descripción del requisito de usuario **RU-01**

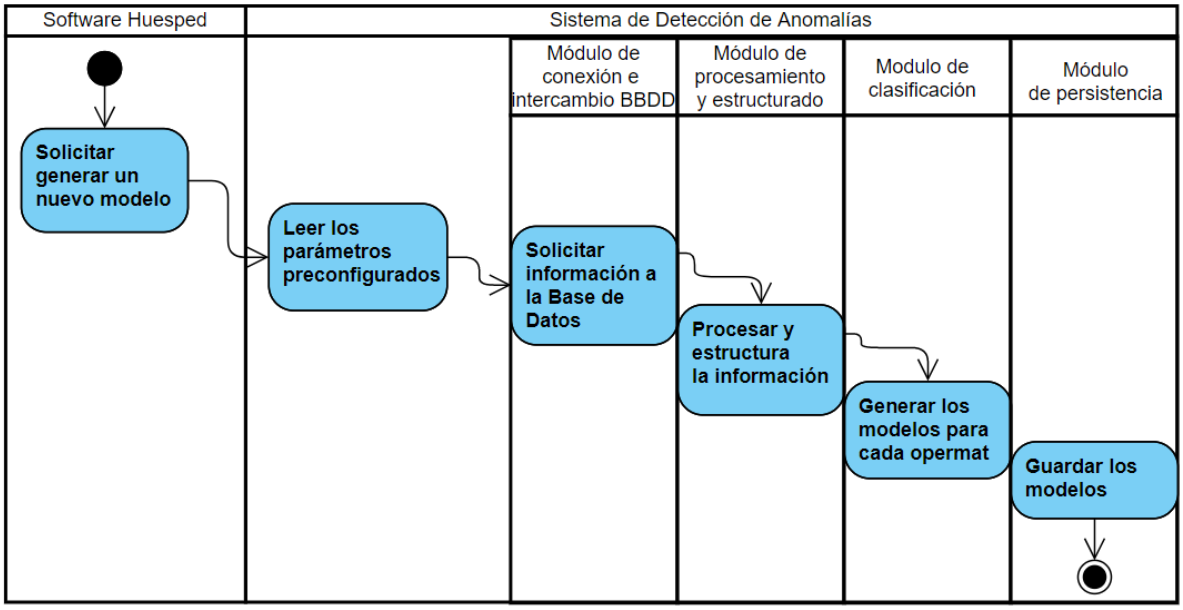


Figura 7.5: Diagrama de actividad correspondiente al caso de uso **RU-01**

RU-02	**Evaluar el comportamiento de la máquina fresadora en vivo**
Actor(es)	Software Huésped (**ACT-01**) y Base de Datos (**ACT-03**)
Descripción	El SDA deberá comportarse como indica el siguiente caso de uso cuando el *Software Huésped* solicite evaluar el comportamiento de la máquina fresadora.
Precondición	Existe un parámetro predefinido denominado valor de tolerancia.
Flujo normal	1. El Software Huésped solicita evaluar el funcionamiento de la máquina fresadora. 2. El sistema lee el parámetro de tolerancia preconfigurado (**RI-04.10**). 3. El sistema solicita a la *Base de Datos* la información relativa a la última entrada (**RI-01**) (**módulo de conexión e intercambio BBDD**). 4. El sistema determina el estado (**RI-07.1**) de la máquina 5. *a*) Si la máquina cambia de estado *apagada* o *sin-actividad* a *en-arranque* o *en-operación* entonces el sistema guarda la última lectura y vuelve al paso 3. *b*) Si la máquina no cambia de estado y esta *en operación* o *en-arranque* entonces añade a las lecturas guardadas la última lectura y vuelve al paso 3. *c*) Si la maquina no cambia de estado y esta *apagada* o *sin-actividad* entonces vuelve al paso 3. *d*) Si la máquina cambia de estado pasando de *en-operación* o*en-material* a otro estado entonces el sistema toma el conjunto de lecturas guardadas y pasa al paso 6. 6. El sistema procesa las entradas guardadas y construye la actividad (**RI-05**) (**módulo de procesamiento y estructurado**). 7. El sistema evalúa la actividad según su Modelo (**RI-06**) correspondiente por *opermat* (**módulo de clasificación**). 8. Si la actividad es considerada *anómala*, el sistema registra la información de la actividad en la base de datos `CNC_Incidencias` (**RI-03**). 9. El sistema registra los cambios sobre el Balance de Situación (**RI-07**) y regresa al paso 3.
Flujos alternos	3. Si la lectura de información falla muestra un mensaje de advertencia y regresa al paso 2. 5. Si el Software Huésped solicita abortar, el caso de uso finaliza.
Postcondición	Modificado el Balance de Estado (**RI-07**).

Tabla 7.11: Descripción del requisito de usuario **RU-02**

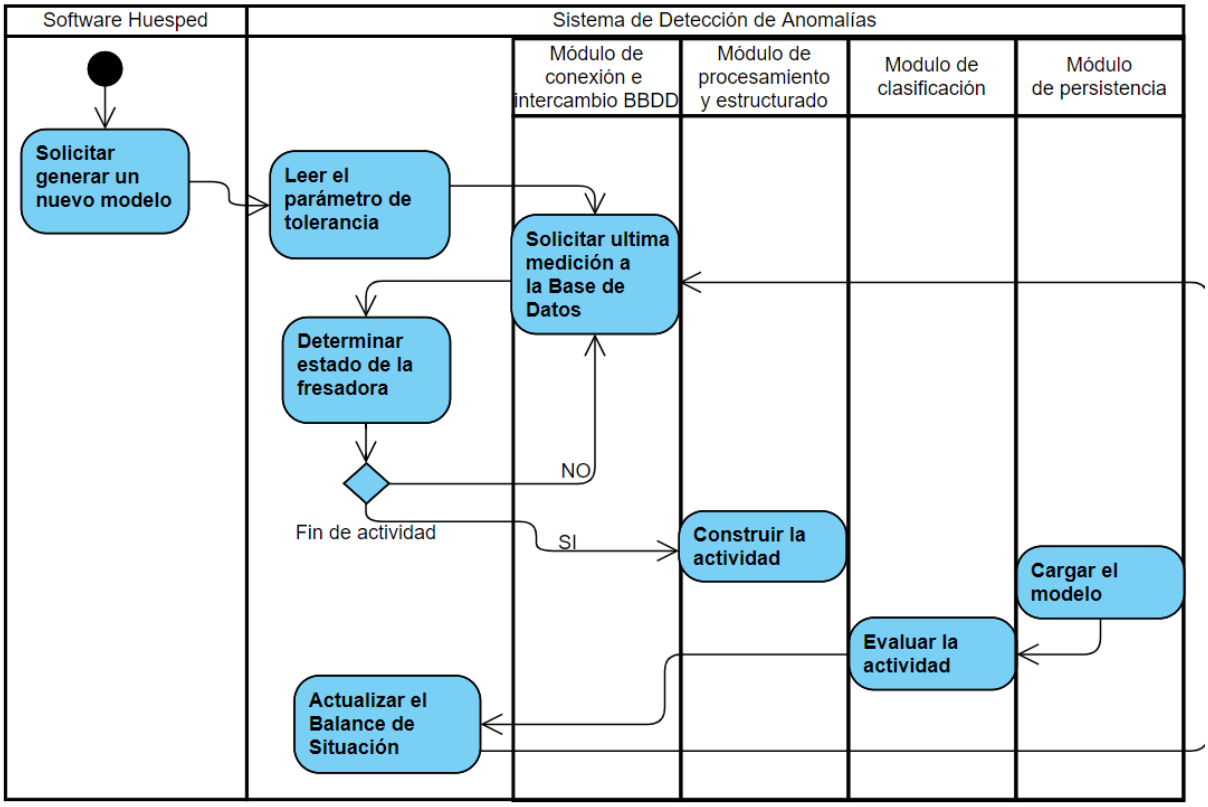


Figura 7.6: Diagrama de Actividad para el caso de uso **RU-02**

RU-03	**Generar un modelo de clasificación con los parámetros modificados por el usuario**
Actor(es)	Usuario (**ACT-02**)
Descripción	El SDA deberá comportarse como indica el siguiente caso de uso cuando el *Usuario* solicite generar un nuevo modelo con unos parámetros distintos a los predefinidos.
Precondición	Existen unos parámetros predefinidos, requeridos en el paso segundo de la secuencia normal
Flujo normal	1. El Usuario solicita crear un nuevo modelo de clasificación. 2. El sistema lee los parámetros preconfigurados: (**RI-04**): fecha inicial (**RI-04.12**), numero de clústeres para la duración (**RI-04.1** y **RI-04.5**), número de clústeres para las series temporales (**RI-04.2** y **RI-04.6**), *opermat* de las actividades de interés a generar el modelo (**RI-04.9**)y duraciones máximas y mínimas de las operaciones (**RI-04.3**, **RI-04.4**, **RI-04.7** y **RI-04.8**). 3. El sistema muestra los parámetros preconfigurados al usuario y solicita su modificación. 4. El Usuario elige e introduce los parámetros que desee modificar. 5. Se sistema modifica los parámetros preconfigurados y ejecuta el caso de uso **RU-01** 6. El caso de uso finaliza satisfactoriamente.
Flujos alternos	5. Si alguno de los parámetros introducidos por el usuario no es válido, el sistema informa del error y vuelve a solicitar dicho parámetro. Cuando todos los parámetros sean correctos, regresa al paso 5.
Postcondición	Generado el modelo de clasificación para los arranques (denominado `modelo_arranque`) y para las operaciones (`modelo_operaciones`).

Tabla 7.12: Descripción del requisito de usuario **RU-03**

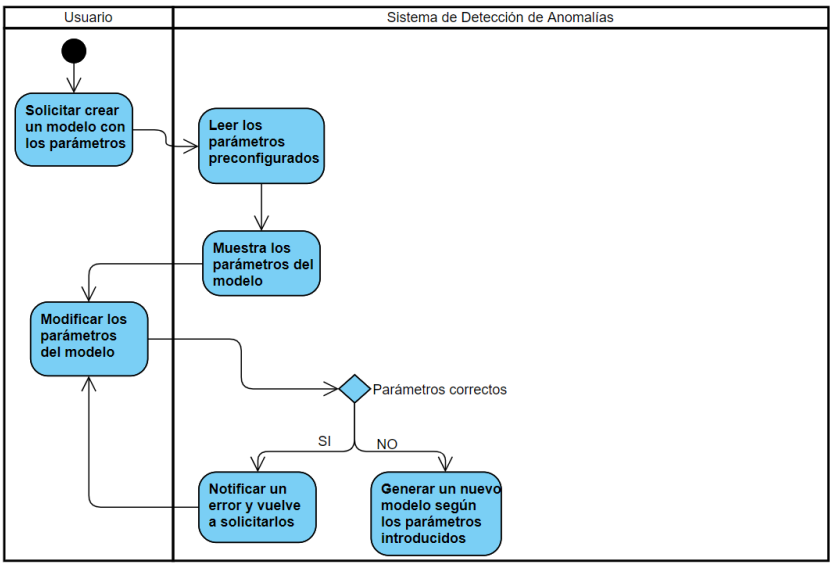


Figura 7.7: Diagrama de actividad para el caso de uso **RU-03**

RU-04	**Visualizar el estado actual de la máquina fresadora**
Actor(es)	Usuario (**ACT-02**)
Descripción	El SDA deberá comportarse como indica el siguiente caso de uso cuando el *Usuario* solicite visualizar la situación actual de la máquina fresadora.
Precondición	Existe un Balance de Situación (**RI-07**).
Flujo normal	1. El Usuario solicita visualizar la información sobre el estado actual de la máquina fresadora. 2. El sistema ejecuta en paralelo el caso de uso **RU-02**. 3. El sistema lee el Balance de Situación (**RI-07**). 4. El sistema muestra el Balance de Situación al usuario. 5. El sistema espera 1 segundo y regresa al paso 2.
Flujos alternos	4. El usuario solicita cerrar la aplicación. 5. El sistema finaliza la ejecución del caso de uso **RU-02** y el caso de uso finaliza satisfactoriamente.
Postcondición	-

Tabla 7.13: Descripción del requisito de usuario **RU-04**

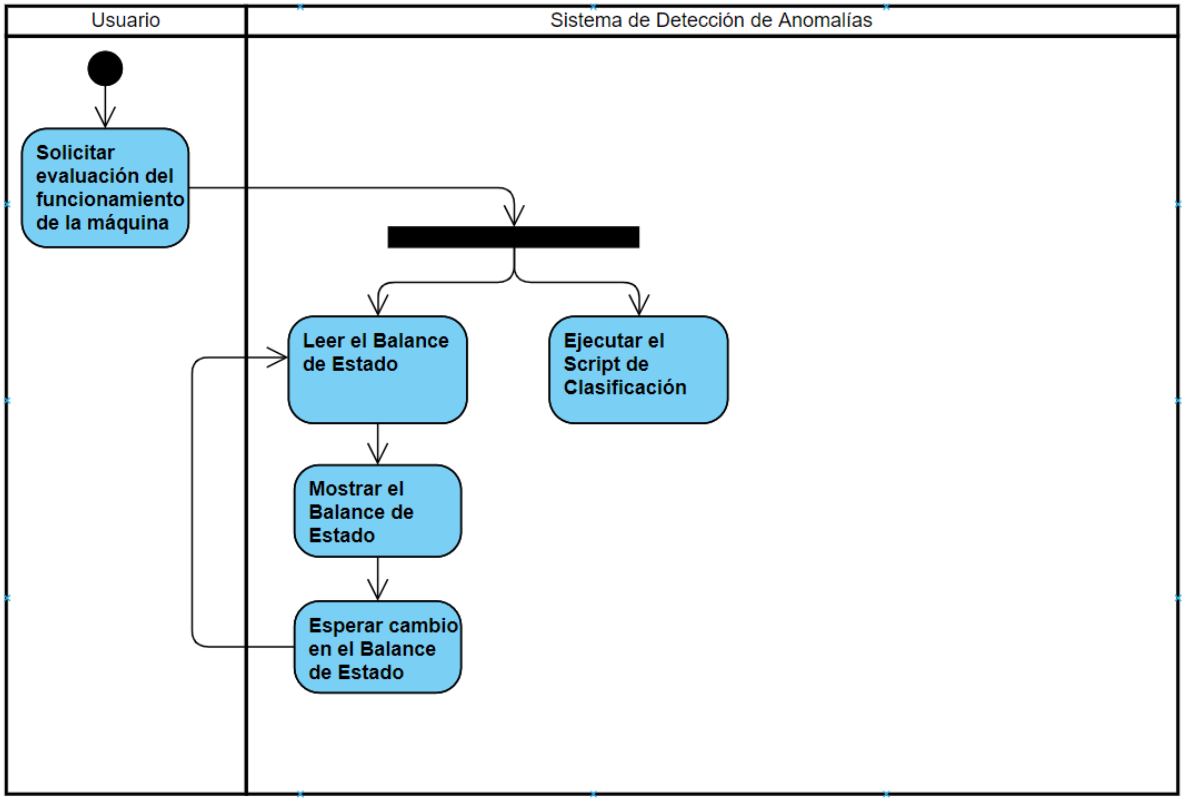


Figura 7.8: Diagrama de actividad para el caso de uso **RU-04**

RU-05	**Realizar el estudio experimental de la configuración para la clasificación**
Actor(es)	Usuario (**ACT-02**)
Descripción	El SDA deberá comportarse como indica el siguiente caso de uso cuando el *Usuario* solicite realizar el estudio experimental para determinar la mejor configuración para la clasificación.
Precondición	-
Flujo normal	1. El Usuario solicita realizar el estudio experimental de la configuración para la clasificación. 2. El sistema solicita al Usuario la información necesaria para realizar el estudio experimental: fuente de datos, operación, material, fecha inicial, duraciones mínima y máxima permitidas, rango de clústeres para la duración y rango de clústeres para la clasificación por k-medias bajo DTW. 3. El sistema toma los datos de la fuente dada y los procesa construyendo las Actividades (**RI-05**) (**módulo de procesamiento y estructurado**). 4. El sistema construye un Modelo (**RI-06**) para cada configuración en el rango introducido (**módulo de clasificación**). 5. El sistema calcula la distancia intra-clúster para cada Modelo y lo representa. 6. El sistema calcula los tiempos requeridos para entrenar cada modelo para cada configuración y los representa. 7. El caso de uso finaliza satisfactoriamente
Flujos alternos	3. Si se produce un problema con la fuente de datos, el sistema lo notifica y finaliza el caso de uso.
Postcondición	-

Tabla 7.14: Descripción del requisito de usuario **RU-05**

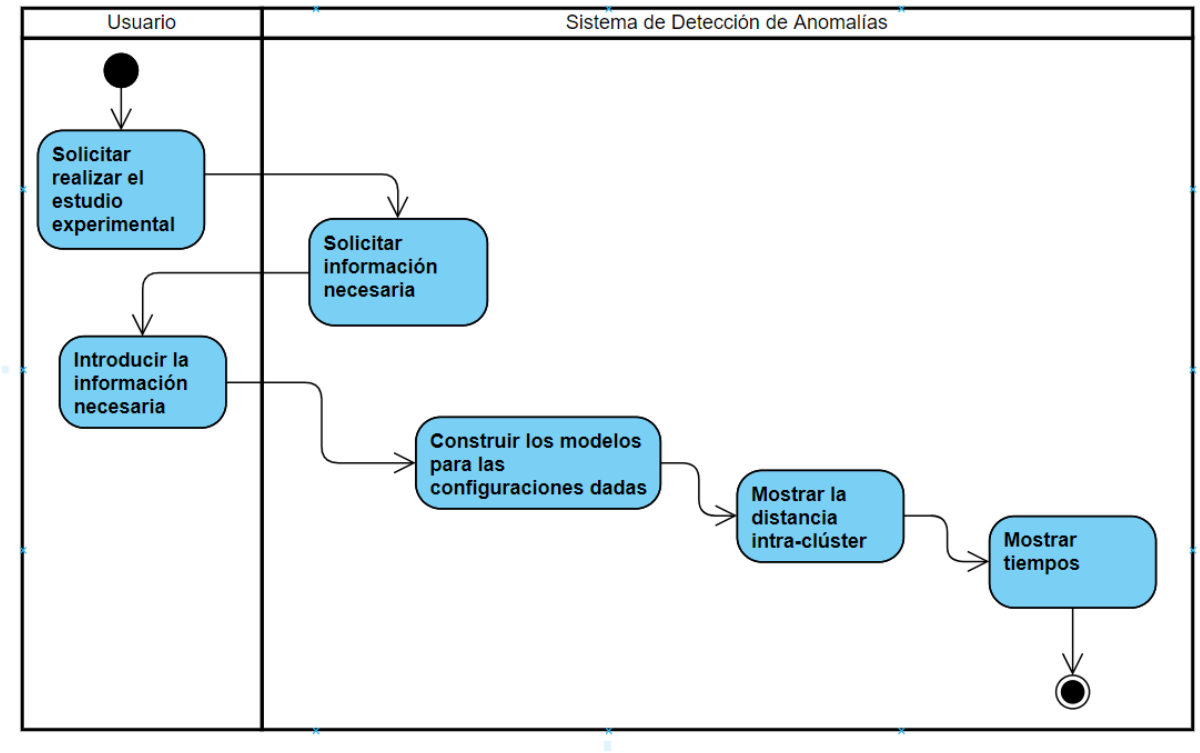


Figura 7.9: Diagrama de actividad para el caso de uso **RU-05**

RU-06	**Realizar el estudio experimental de la configuración para la clasificación**
Actor(es)	Usuario (**ACT-02**) y Base de Datos.
Descripción	El SDA deberá comportarse como indica el siguiente caso de uso cuando el *Usuario* solicite realizar el estudio experimental para determinar la mejor configuración para la clasificación.
Precondición	-
Flujo normal	1. El Usuario solicita realizar el estudio experimental para la tolerancia. 2. El sistema solicita al Usuario la información necesaria para realizar el estudio experimental: operación y material (*opermat*). 3. El sistema toma los datos de la Modelos construidos (**RI-06**) (**Módulo de persistencia**) para cada valor de *opermat* introducido. 4. El sistema representa el porcentaje de elementos presentes en cada clúster de cada Modelo según el valor de *opermat*. 5. El sistema solicita al usuario un valor de candidato a tolerancia. 6. El Usuario introduce el valor pedido 7. El sistema representa el número de clústeres anómalos y usuales para el candidato a valor de tolerancia, discriminando por *opermat*. 8. El caso de uso finaliza satisfactoriamente
Flujos alternos	3. Si se introduce un *opermat* no valido, el sistema notifica al error y regresa al paso 2. --- 7. Si se introduce un candidato a tolerancia no valido, el sistema notifica al error y regresa al paso 2.
Postcondición	-

Tabla 7.15: Descripción del requisito de usuario **RU-06**

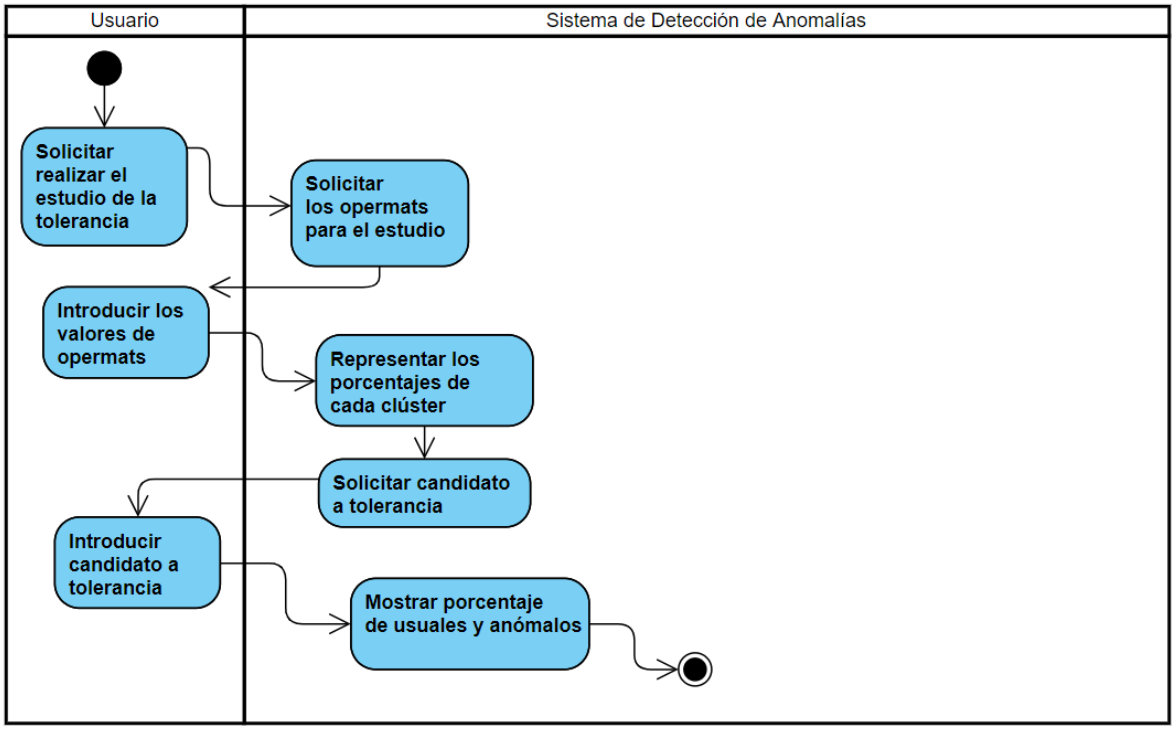


Figura 7.10: Diagrama de actividad para el caso de uso **RU-06**

7.3. Diseño

En esta sección se recogen todos los detalles relativos al diseño del sistema de detección de anomalías, en cuanto a la estructuración de la información y del modelo de clasificación. Se opta por un diseño orientado a objetos donde se implementan diferentes clases para estructurar la información y para definir la arquitectura del modelo descrito en la *Sección 5.3*

7.3.1. Representación lógica de la información

Con objeto de estructurar adecuadamente la información, cargada desde la base de datos como un `pandas.DataFrame`, identificando y separando las actividades (operaciones o arranques) realizadas. Cada actividad realizada es estructurada como una instancia de la clase `Actividad` y el conjunto de actividades es estructurado mediante la clase `Conjunto_actividad`.

La clase **`Actividad`** está compuesta por diferentes atributos que recogen la información almacenada sobre cada actividad. Así, la información recogida por los sensores se almacena como una serie temporal sin etiqueta de tiempo (una `lista` en *Python*), y se mantiene la información sobre el tipo de operación, material y herramienta (según los códigos recogidos en las *Tablas 3.1, 3.2 y 3.3*). Además, se crea un identificador único para cada tipo de actividad, denominado *opermat*, y que es el resultado de multiplicar por 10 el código de la operación y sumarle el código del material. Los arranques reciben un *opermat* de -10 (a los arranques se les asigna -1 como clave de operación). Los atributos de la clase `Actividad` son:

- *`corriente1:`* serie temporal (lista) con las mediciones realizadas cada segundo por el sensor de corriente de fase 1.
- *`corriente2:`* serie temporal (lista) con las mediciones realizadas cada segundo por el sensor de corriente de fase 2.
- *`corriente3:`* serie temporal (lista) con las mediciones realizadas cada segundo por el sensor de corriente de fase 3.
- *`acelX:`* serie temporal (lista) con las mediciones realizadas cada segundo por el sensor acelerómetro-eje-z
- *`acelY:`* serie temporal (lista) con las mediciones realizadas cada segundo por el sensor acelerómetro-eje-z
- *`acelZ:`* serie temporal (lista) con las mediciones realizadas cada segundo por el sensor acelerómetro-eje-z
- *`operacion:`* número entero con el código que representa el tipo de operación realizada.
- *`material:`* número natural con el código que representa el material utilizado.
- *`opermat:`* número entero con el código que representa el *opermat* asignado.
- *`duración:`* número natural que se corresponde la duración total de la actividad

- *fecha_ini:* fecha completa (día y hora exactos) de comienzo de la actividad.
- *día:* día en el que se realizó la actividad
- *orden_fab:* orden de fabricación (plano) asociada a la pieza.
- *herramienta:* número natural con el código que representa la herramienta utilizada.

Por otro lado, la clase **`Conjunto_actividad`** se compone de un único atributo, `lista_actividad`, que, como su propio nombre indica, es una lista de instancias de la clase `Actividad`. Así, los métodos de esta clase están destinado a facilitar la utilización y extracción de información concreta sobre las actividades realizadas y que han agrupadas bajo la propia clase. Entre estos, destacan los siguientes métodos:

- *llenadoYprocesado:* dado un `pandas.DataFrame` con la información extraída de la base de datos sobre el funcionamiento y dado el conjunto de índices donde comienza y termina cada actividad, extrae las actividades realizadas, creando una instancia de la clase `Actividad` para cada una y las almacena en el atributo `lista_actividad`.
- *obtener_recientes:* dado n un entero positivo, crea una nueva instancia de `Conjunto_actividad` con las n actividades más recientes.
- *depurado_por_duracion/(*resp. *depurado_por_duracion_max):* dado n un entero positivo, crea una nueva instancia de `Conjunto_actividad` donde se han suprimido las actividades con duración inferior (resp. superior) a n.
- *depurar_ceros:* crea una nueva instancia de `Conjunto_actividad` donde se han suprimido todas las actividades con operación o material cero (arranques incluidos). Es útil para eliminar los errores en el etiquetado.
- *extraer_por_opermat (*resp. *extraer_por_operacion* o *extraer_por_material):* dado un identificador de *opermat* (resp. operación o material) crea una nueva instancia de `Conjunto_actividad` donde todas las actividades son de dicho *opermat* (resp. operación o material).

7.3.2. Arquitectura lógica del modelo

En la *Sección 5.3* se describe la arquitectura del modelo de clasificación elegido. Este modelo es un enfoque híbrido donde primero se clasifica por la duración de cada serie temporal (k-medias tradicional), y seguidamente se clasifica por la propia serie temporal (k-medias bajo la DTW).

En primer lugar, con objetivo de dar independencia completa al modelo de clasificación del tipo de representación elegida para la información, y así facilitar su reutilización como módulo independiente en otros proyectos, se crea la clase **`Dato_generico`**. Dicha clase consta de solo tres atributos:

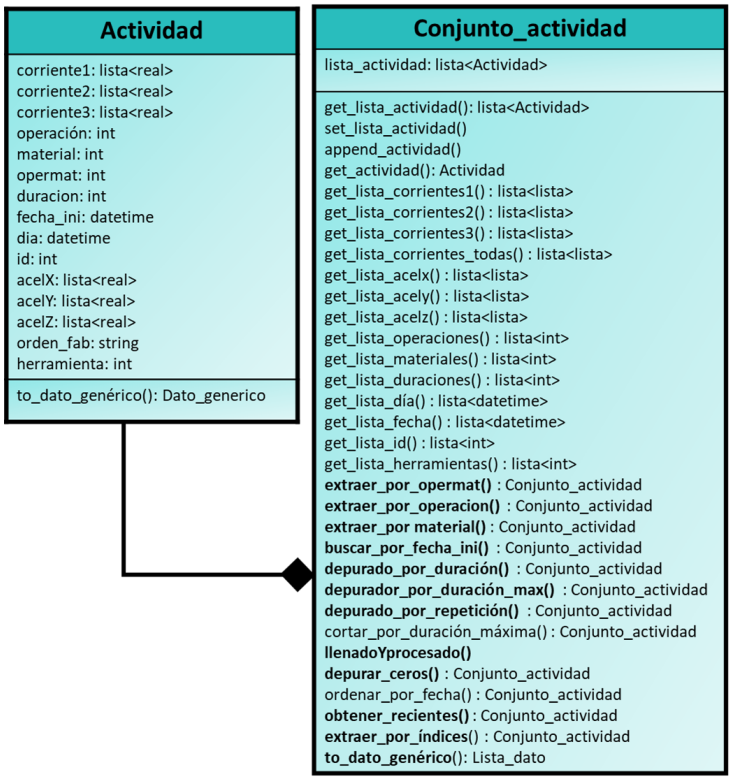


Figura 7.11: Diagrama de clases para la estructuración de las actividades.

- *tipo:* permite discriminar entre diferentes categorías de tipo de información, de forma que para cada categoría distinta se va a crear un modelo diferente
- *valor:* serie temporal (lista) de longitud y dimensión arbitraria, que representa la información utilizada para la clasificación
- *fecha:* fecha en la cual se tomó la medición. Se utiliza para discriminar las observaciones más antiguas cuando sea necesario.

De forma análoga al estructurado de las actividades, se diseña la clase **`Lista_dato`**, cuyo único atributo `lista` es una agrupación de diferentes instancias de la clase `Dato_generico`. Así, esta clase tiene el objetivo de reunir diferentes instancias de la clase `Dato_generico` y dotar de un conjunto de métodos que faciliten su utilización. En particular destacan:

- *obtener_recientes:* dado un número natural n, crea una nueva instancia de `Lista_dato` con n objetos de tipo `Dato_generico` cuya *fecha* sea más reciente.
- *extraer_por_tipo:* dado un identificador de algún tipo (ID) de dato predefinido crea una nueva instancia de `Lista_dato` donde todas las instancias tienen por atributo `tipo` el ID.

En consecuencia, cada objeto de la clase `Actividad` debe ser transformado en un objeto de la clase `Dato_generico` para su clasificación por el modelo. Para ello, se implementa el método `to_dato_generico` en la clase `Actividad`. Análogamente, los objetos de la clase `Conjunto_actividad` deben ser transformados en la clase `Lista_dato`, para lo que se implementa un método de mismo nombre.

Por otro lado, para implementar el modelo de clasificación con la arquitectura propuesta en la *Sección 5.3* se diseña la clase **`Kmedias_tiempo`**. Esta clase dispone de los siguientes atributos:

- *`max_data:`* número máximo de datos que se utiliza para entrenar el modelo. En caso de que el de datos de entrenamiento sea superior se suprimen las instancias más antiguas.
- *`n_clusters_dur:`* determina el número de clústeres usado para la clasificación por k-medias según la duración de cada serie.
- *`n_clusters_ts:`* determina el número de clústeres usado para la clasificación por k-medias de las series temporales bajo la distancia DTW.
- *`km_dur:`* resultado de clasificar las duraciones de las series temporales por el algoritmo de k-medias tradicional (como instancia de `sklearn.cluster.KMeans`).
- *`km_dic:`* diccionario de datos donde las claves son los identificadores de los clústeres generados por la clasificación de la distancia (`km_dur`) y los valores son el resultado de clasificar cada subconjunto de series temporales obtenido en la clasificación por duración mediante k-medias bajo la DTW. Es decir, el valor dado para cada clave es un objeto de la clase `tslearn.clustering.TimeSeriesKMeans`.
- *`data:`* conjuntos de los datos que se han usado para entrenar el modelo.

Además, esta clase dispone de tres métodos fundamentales:

- *`entrenar:`* dado un conjunto de datos de entrenamiento como objeto de la clase `Lista_dato`, entrenar el modelo según el esquema de la *Figura 5.8* dando valor a los atributos de `km_dur` y km_dic.
- *`evaluar:`* dado un dato test como instancia de la clase `Dato_generico`, clasifica la frecuencia dicho dato según el modelo. El procedimiento de evaluación es análogo al de entrenamiento del modelo:
  1) En primer lugar se evalúa según la duración (representado por `km_dur`) y se asigna a un clúster de la división por duración denotado por $\mathcal{D}_c$.
  2) Se extrae del diccionario `km_dic` la clasificación por k-medias (`TimeSeriesKMeans`) correspondiente para el clúster asignado a la duración c y se evalúa por dicho modelo. Como resultado se obtiene un clúster $K_{c,j}$ al que ha sido asignada la serie del objeto a evaluar.

3) El resultado de la evaluación es número elementos presentes en el clúster asignados y el número de elementos del conjunto de entrenamiento que sean del mismo tipo que el dato clasificado.

- *reentrenar:* recibe un conjunto de datos de entrenamiento como objeto de la clase `Lista_dato`, y entrena (método `entrenar`) el modelo con los nuevos datos y con los datos antiguos (atributo `data`) discriminando los más antiguos cuando se supere el máximo marcado (atributo `max_data`).

Por último, cada tipo de *opermat* se desea construir un modelo de clasificación propio (es decir, diferentes objetos de la clase `Kmedias_tiempo`). Para administrar adecuadamente los diferentes modelos y unificarlos se crea la clase **`Agrupacion_mis_kmedias`**. Dicha clase tiene un único atributo, denominado *`diccionario`*, que es una estructura de datos tipo diccionario en la cual las claves son los diferentes tipos de `Dato_generico` (es decir, los diferentes valores de *opermat* seleccionado) y su valor asociado es un objeto `Kmedias_tiempo` que representa el modelo de clasificación para dicho tipo de dato (*opermat*).

Además, esta clase dispone de los métodos *`entrenar`* y *`reentrenar`* que dado un conjunto de datos como `Lista_dato`, separa cada tipo de dato y entrena o reentrena respectivamente los modelos correspondientes. También dispone del método , *`evaluar`* que dada una muestra como `Dato_generico` realiza su evaluación mediante el método homónimo del modelo de clasificación correspondiente por el tipo de dato.

En la siguiente figura (*Figura 7.13*) se presenta el diagrama de clases completo para el sistema de detección de anomalías.

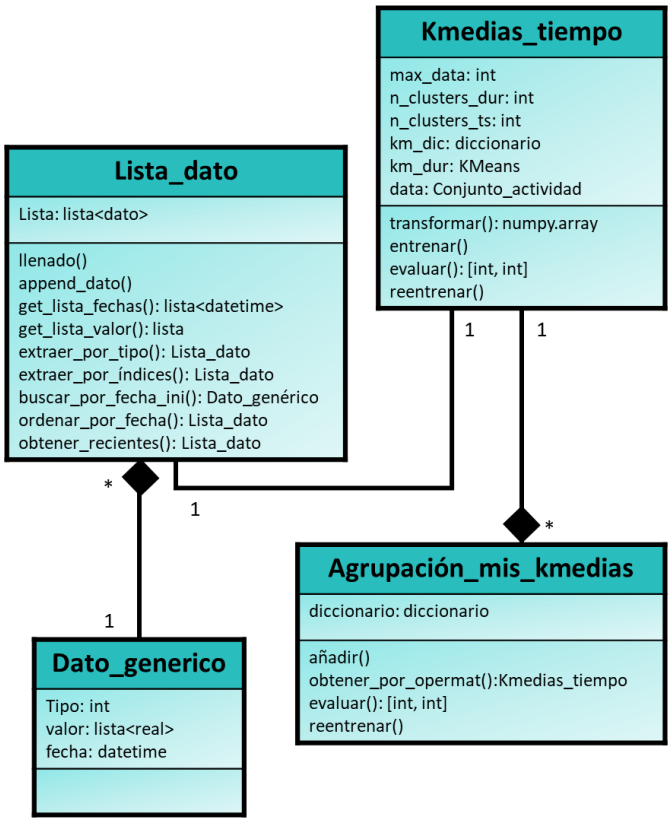


Figura 7.12: Diagrama de clases para el diseño del modelo de clasificación

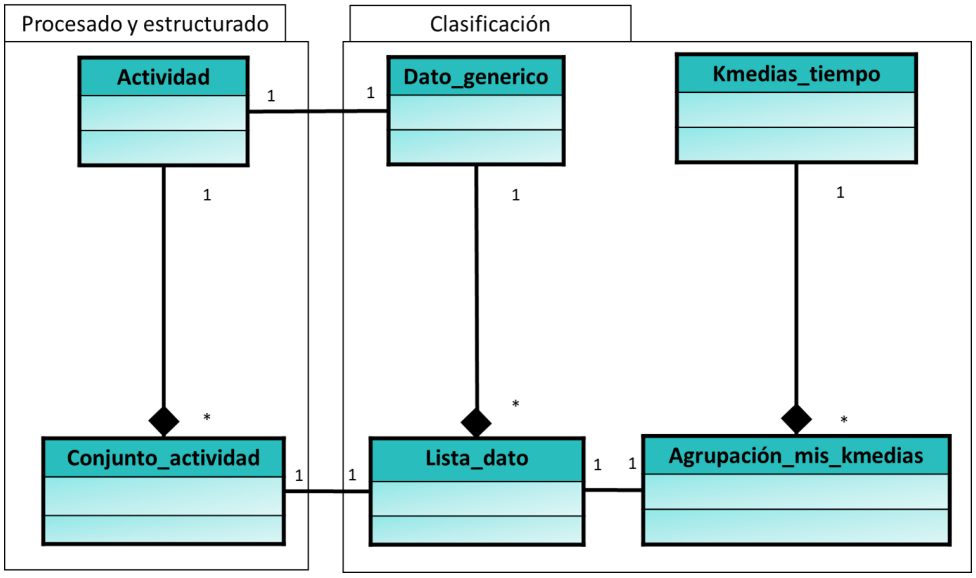


Figura 7.13: Diagrama de clases completo del sistema de detección de anomalías

7.4. Reglas de Negocio

En esta sección se describen todas las políticas, estándares y regulaciones que restringen algún aspecto y afecta al Sistema de Detección de Anomalías. Las reglas de negocio están recogidas en la *Tabla 7.16*.

Id	Regla de Negocio
RN-01	La estructura de las tablas en las que se recogen las mediciones (**RI-01** y **RI-02**) de los sensores es ajena al Sistema de Detección de Anomalías y su estructura esta prefijada.
RN-02	La estructura de la tabla de registro de anomalías (**RI-03**) y de la información almacenada en dicha tabla queda determinada de forma externa al diseño del sistema y depende de el módulo externo *Grafana*.
RN-03	El Sistema de Clasificación deberá tener la capacidad para ejecutarse de forma autónoma en un servidor independiente realizando exclusivamente la acción de evaluación de las actividades.

Tabla 7.16: Reglas del negocio

7.5. Atributos de calidad

Los atributos de calidad recogen aquellas características no funcionales deseables para el Sistema de Detección de Anomalías. Por tanto, dada la naturaleza del sistema en desarrollo y los objetivos propuestos, los atributos de calidad considerados estan recogidos en la *Tabla 7.17*.

Id	Tipo	Descripción
Cal-01	Robustez	El sistema sera capaz de detectar fallos en la conexión con la base de datos, para volver a conectar y ejecutar la consulta correspondiente.
Cal-02	Integridad	El sistema solo considerará las operaciones que estén recogidas doblemente en las tablas `CNC_Funcionamiento` y `tiempoOperaciones`.
Cal-03	Integridad	El sistema ignorará aquellas actividades cuya duración sea superior a 6 horas por ser consideradas imposibles y asociadas a errores en el etiquetado.
Cal-04	Robustez	El sistema será capaz de recuperarse frente a archivos corruptos o con una estructura no adecuada, notificando al usuario dicho error.
Cal-05	Usabilidad	El sistema será fácilmente usable y dispondrá de una interfaz amigable para el usuario final.

Tabla 7.17: Atributos de calidad

Capítulo 8

Desarrollo del proyecto

En este capítulo se detalla el desarrollo temporal y asignación de actuaciones seguido durante el proyecto. En el capítulo anterior (*Capítulo 7*) se han recogido todos los requisitos de uso, actores, etc. Sin embargo, las funcionalidades necesarias no han sido conocidas desde el comienzo del proyecto, sino que han ido surgiendo según se avanzaba. Es más, al comienzo del proyecto no se conocía que técnicas de clasificación y de representación eran las más adecuadas, que se eligieron tras familiarizarse con los datos y comprender el problema (lo que se denominara etapa 0).

Se sigue un ciclo de vida evolutivo con 3 etapas. En cada una de las fases se desarrolla e implementa completamente diferentes funcionalidades que componen los módulos del proyecto. Es más, en cada etapa se desarrollan completamente algunos de los módulos, sin excluir que en las etapas futuras se requieran añadir nuevas funcionalidades secundarias. Sin embargo, las funcionalidades añadidas y validadas en una etapa no pueden ser modificadas en las etapas posteriores.

Este ciclo de vida se elige por la naturaleza del proyecto: alta variabilidad de requisitos, un entorno cambiante con funcionalidades desconocidas al comienzo del proyecto y desconocimiento del modelo de clasificación determinado por el propio proyecto en alguna de sus fases. Además, este proyecto esta caracterizado por disponer de una importante realimentación por parte del "*product owner*"[1], que permite completar cada fase con su *feedback* y, puntualmente, da una retroalimentación hacia las fases anteriores.

Cada una de las etapas siguientes tiene sus propios requisitos tanto de usuario como funcionales, y busca satisfacer alguno de los objetivos. En esta sección se incluye el diseño, implementación y pruebas de cada fase, haciendo una referencia a la parte de análisis y diseño correspondientes recogidos en el *Capítulo 7*. Las etapas en las que se divide el proyecto son:

Etapa Cero: Familiarización y elección de un algoritmo. Antes de comenzar con el análisis de la aplicación, se requiere estudiar el problema y elegir tanto la técnica para estructurar la información como el algoritmo de clasificación adecuado al problema. En esta fase no se desarrolla ningún requisito de usuario, pues no se da ninguna funcionalidad final

[1] En el sentido de un equipo *Scrum*: persona encargada de maximizar el valor del trabajo, y que representa los intereses e ideas del cliente.

Etapa Primera: El *Script* de Clasificación. Se comienza desarrollando las funcionalidades asociadas al subsistema principal, dadas por los requisitos de usuario **RU-01** y **RU-02**.

Etapa Segunda: Interfaces para el *Script* de Clasificación. Se diseñan e implementan las interfaces para visualizar el Balance de Situación de la máquina fresadora (**RU-04**) y para la construcción de un modelo con parámetros distintos a los predefinidos (**RU-03**).

Etapa Tercera: El estudio experimental. Se desarrollan las funcionalidades que permiten realizar el estudio experimental para elegir la mejor configuración para el modelo.

A su vez, cada una de las etapas anteriores se descompondrá en ciertas fases, de forma que en cada fase se dedica principalmente alguno de los módulos del Sistema de Detección de Anomalías (*Sección 6.3*), sin excluir que sea necesario añadir funcionalidades en fases posteriores, especialmente dentro de la misma etapa. Además, toda etapa tiene una fase de pruebas que retroalimenta las etapas anteriores. El resultado de cada etapa es un sistema completamente funcional, probado y validado.

8.1. Etapa cero

En la etapa cero se busca estudiar el problema "búsqueda de patrones y detección de anomalías" con el histórico de datos disponible. Así, se busca cumplir el objetivo **Obj-01**, para lo que se requiere procesar, estructurar y analizar la información disponible (**Obj-02.1**, **Obj-02.2**). Como resultado final se elige una técnica de clasificación que se adecúe al problema y a la estructura de los datos (**Obj-02.3**).

Para el cumplimiento de estos objetivos se diseña e implementa el **módulo de procesamiento y estructurado**, cuyo esquema conceptual se recoge en la *Sección 7.3.1*, con las funcionalidades que se describen en dicha sección y que permiten estructurar las Actividades (**RI-05**). La información por estructurar se encuentra en un documento `csv` tabulada y se carga como un objeto `pandas.DataFrame`.

Además, con objeto de almacenar de forma persistente la información de las Actividades (**RI-05**) procesada, en esta etapa se diseña e implementa el **módulo de persistencia**, con los métodos que permiten empaquetar y guardar un objeto de una clase arbitraria como un fichero binario `.pickle`, y recuperar el objeto de dicho fichero.

Esta etapa inicial no pertenece al desarrollo del Sistema de Detección de Anomalías, sino que lo fundamenta. Los resultados de esta etapa se han recogido en la *Sección 5.1*, que han justificado la elección de la técnica de clasificación y la arquitectura del modelo expuestos en la *Sección 5.3*. En resumen, se opta por modelizar las operaciones como series temporales y clasificarlas por un modelo de *clustering* híbrido.

8.1.1. Requisitos Funcionales

Los requisitos funcionales desarrollados en esta etapa se recogen en la *Tabla 8.1*.

Id	Requisitos funcionales
RF-01	El sistema procesará la información tabulada recibida como un objeto `pandas.DataFrame` y la dispone como Actividades (**RI-05**), estructuradas por las clases `Actividad` y `Conjunto_actividad`.
RF-02	El sistema guardará una instancia de una clase arbitraria como un fichero binario con extensión `.pickle`.
RF-03	El sistema extraerá y reconstruirá la instancia de una clase dada almacenada en un fichero binario `.pickle`.
RF-04	El sistema leerá la información contenida en un archivo `csv` y la cargará como clase `pandas.DataFrame`

Tabla 8.1: Requisitos Funcionales de la Etapa 0

8.1.2. Detalles de la implementación

Para la implementación se utilizan los módulos `pandas` y `numpy` para la manipulación de los datos recogidos por los sensores y estructurados en un `csv`. También se utiliza el módulo `datetime` para el manejo de las fechas y el módulo `pickle` para la serialización y deserialización de objetos de clases.

Los módulos *Python* creados son `ClasesActividades.py` y `FuncionesPropias.py`. El primero de los archivos contiene la implementación de las clases `Actividad` y `Conjunto_actividad`, y el segundo contiene las funciones que permiten transformar la información leída de un `csv` en instancias de las clases anteriores. Por otro lado, el módulo de persistencia se implementa como una "clase" sin atributos denominada `GuardaCarga` y definida en el archivo `FuncionesPropias.py`.

8.2. Etapa primera

En la primera etapa se diseña e implementa el *Script* de Clasificación. Con ello se satisfacen los objetivos **Obj-02.4** y **Obj-03**, implementando los casos de uso **RU-01** y **RU-02**. Esta etapa se desarrolla en las siguientes fases:

Fase 1: Construcción del Modelo de Clasificación. Se construye el modelo de clasificación (**RI-06**) cuya arquitectura se definió en la *Sección 5.3* y cuyo diseño se presentó en la *Sección 7.3.2*, modelado por cuatro clases: `Dato_generico`, `Lista_dato`, `Kmedias_tiempo` y `Agrupacion_mis_kmedias`.

Fase 2: Implementación del Depurado sobre las actividades. Se diseñan e implementan los métodos que permiten depurar el conjunto total de operaciones (representados

como clase `Conjunto_actividad`) por su duración, valor de *opermat* y fecha.

Fase 3: Construcción del módulo de conexión con la Base de Datos. Se implementan las funcionalidades que permiten ejecutar una consulta sobre la Base de Datos SQL externa, y cargar el resultado de dicha consulta como un objeto `pandas.DataFrame`. También se implementan las funcionalidades que permiten insertar un objeto `pandas.DataFrame` en una tabla `SQL`, siempre que ambos presenten la misma estructura de columnas.

Fase 4: *Script* para la construcción del Modelo. Se implementa completamente el caso de uso **RU-01**, desarrollando las funcionalidades que permiten construir el Modelo de Clasificación (**RI-06**) a partir de las Configuraciones predefinidas (**RI-04**).

Fase 5: Desarrollo de las funcionalidades que permiten la valoración en vivo. Se diseña e implementa el *Script* que cumple el caso de uso **RU-02**, capaz de determinar el Balance de Situación (**RI-06**) de la máquina fresadora y detectar y evaluar las actividades. Para esto último se añaden nuevas funcionalidades al **módulo de procesamiento y estructurado** tales que dada una actividad como un objeto `pandas.DataFrame` lo transforme en un objeto de la clase `Actividad`.

8.2.1. Requisitos Funcionales

Los requisitos funcionales desarrollados en esta etapa se recogen en la *Tabla 8.2*.

8.2.2. Detalles de la implementación

Para la implementación de las funcionalidades asignadas a esta etapa se han utilizado los siguientes módulos *Python*:

- Para construir el Modelo de Clasificación se han utilizado los módulos `tslearn` y `sklearn`. En particular, destacan las clases `sklearn.cluster.KMeans` (algoritmo de k-medias en la versión tradicional, necesario para clasificar las duraciones), `tslearn.clustering.TimeSeriesKMeans` (algoritmo de k-medias bajo la DTW aplicable a series temporales) y `tslearn.metrics.dtw` (implementación de la distancia 2–DTW).
- Para la conexión lógica con la base de datos, ejecución de consultas e inserción de información se han utilizado los módulos `pyodbc` y `sqlalchemy`.
- Para el suavizado de las series temporales mediante el algoritmo de medias móviles en su versión centrada y aplicando las hipótesis del modelo aditivo se ha utilizado el método `seasonal_decompose` del módulo `statsmodels.tsa.seasonal`.

Id	Requisitos funcionales
RF-05	El sistema leerá los parámetros preconfigurados (**RI-04**) contenidos en un archivo de *Python*.
RF-06	El sistema solicitará a la base de datos la información de todas las Mediciones (**RI-01**) posteriores a una fecha dada.
RF-07	El sistema solicitará a la base de datos la información de todas las Realizaciones (**RI-02**) posteriores a una fecha dada.
RF-08	El sistema determinará el valor de *opermat* (**RI-05.9**) de cada Actividad (**RI-05**).
RF-09	El sistema separará las actividades por su valor de *opermat* (**RI-05.9**).
RF-10	El sistema generará un modelo para cada tipo de *opermat* de interés (**RI-04.9**)
RF-11	El sistema tendrá la capacidad de detectar los errores en la conexión con la base de datos y notificar dicho error al usuario.
RF-12	El sistema solicitará a la base de datos la información relativa a la última Medición (**RI-01**).
RF-13	El sistema tendrá la capacidad de determinar el estado (**RI-07.1**) de la máquina fresadora.
RF-14	El sistema tendrá la capacidad de detectar cuando se producen cambios en el estado (**RI-07.1**) de la máquina fresadora
RF-15	El sistema almacenará las Mediciones (**RI-01**) extraídas de la base de datos (**RF-12**).
RF-15	El sistema estructurará una Actividad (**RI-05**) dada de forma tabulada.
RF-16	El sistema evaluará la Actividad (**RI-05**) según el Modelo de Clasificación (**RI-06**) correspondiente.
RF-17	El sistema notificará cuando una Actividad (**RI-05**) no tenga Modelo (**RI-06**) asignado.
RF-18	El sistema registrará todas aquellas Actividades (**RI-05**) consideradas anómalas por el Modelo (**RI-06**) como Incidencias (**RI-03**) en la base de datos.
RF-19	El sistema registrará los cambios sobre el Balance de Situación (**RI-07**).

Tabla 8.2: Requisitos Funcionales de la Etapa 1

Para implementar las funcionalidades correspondientes, se han creado los siguientes módulos *Python*:

- El Modelo de Clasificación se ha implementado en el archivo `ModuloClasificacion.py`.
- La conexión con la base de datos se ha implementado en el archivo `conectarBBDD.py`, y la información necesaria para dicha conexión (nombre de la base de datos, usuario, etc.) así como las consultas a ejecutar se encuentran en el fichero `configBBDD.py`
- Las funcionalidades que permiten generar el Modelo están implementadas en el fichero `generador_del_modelo`.
- El *script* que permite evaluar actividades en vivo a partir de un modelo es `main.py`.
- Las configuraciones predefinidas y necesarias para el correcto funcionamiento se encuentran contenidas en `configuracion.py`.

8.2.3. Validación y pruebas

Para validar el correcto comportamiento del *Script* de clasificación se realizan dos pruebas de caja negra, recogidas en las *Tablas 8.3 y 8.4*.

PCN-01	Generación del modelo
Prerrequisitos	-
Acciones	El Software Huésped solicita generar el modelo de clasificación según los parámetros
Resultado esperado	Se generan los modelos de clasificación para las operaciones y para los arranques, guardados como archivo `pickle` y denominados `modelo_arranque.pickle` y `modelo_operaciones.pickle`
Resultado obtenido	Se tiene el resultado esperado
Valoración de la prueba	Superada con éxito

Tabla 8.3: Prueba de caja negra sobre la generación de los modelos de clasificación

8.3. Etapa segunda

En la segunda etapa se busca implementar las interfaces que permitan a un Usuario (**ACT-02**) interactuar con el *Script* de Clasificación. Por tanto, se va a desarrollar la primera parte del módulo secundario, implementando los requisitos de usuario **RU-03** y **RU-04**.

PCN-02	Evaluación en vivo
Prerrequisitos	Existen los Modelos de Clasificación para las operaciones y para los arranques
Acciones	El Software Huésped solicita iniciar la evaluación de las actividades en vivo de la máquina fresadora
Resultados esperado	▪ Se realiza la lectura del último dato de la Base de Datos y se analiza el estado ▪ Se detecta que se ha finalizado una actividad y se evalúa según el Modelo de Clasificación correspondiente. ▪ Se registra el cambio en el Balance de Situación en un archivo `json`.
Resultados obtenido	Se tiene el resultado esperado
Valoración de la prueba	Superada con éxito

Tabla 8.4: Prueba de caja negra sobre la evaluación de las actividades

8.3.1. Requisitos Funcionales

Los requisitos funcionales desarrollados en esta etapa se recogen en la *Tabla 8.5*.

Id	Requisitos funcionales
RF-20	El sistema ofrecerá al Usuario (**ACT-01**) la posibilidad de visualizar los parámetros preconfigurados.
RF-21	El sistema mostrará al Usuario (**ACT-01**) los parámetros preconfigurados (**RI-04**).
RF-22	El sistema ofrecerá al Usuario (**ACT-01**) la posibilidad de modificar los parámetros preconfigurados.
RF-23	El sistema registrará los cambios en los parámetros preconfigurados y generará el correspondiente modelo (**RI-06**).
RF-24	El sistema ofrecerá al Usuario (**ACT-01**) la posibilidad de visualizar el Balance de Situación (**RI-07**).
RF-25	El sistema tendrá la capacidad de leer el Balance de Situación (**RI-07**), dado como un archivo `json`.
RF-26	El sistema proporcionará al Usuario (**ACT-01**) la información del Balance de Situación (**RI-07**)
RF-27	El sistema tendrá la capacidad de ejecutar el módulo principal, en paralelo, si este no se está ejecutando.

Tabla 8.5: Requisitos Funcionales de la Etapa 0

8.3.2. Detalles de la implementación

Para la implementación de las interfaces se ha utilizado el *framework* `Dash` (de *Plotly*). Este *framework* permite crear paneles con *Python* puro, que son transformados en aplicaciones web dinámicas. `Dash` está basado en `Flask`, `Plotly` y `ReactJS`, lo que le da unas características adecuadas al proyecto:

- Las aplicaciones se rendericen en un navegador web.
- Sea multiplataforma y posea un diseño adaptativo preparado para móviles.
- Es de código libre (*Opensource*).

En la siguiente sección (8.3.3) se recogen los detalles sobre el diseño de las interfaces. Para implementar las funcionalidades e interfaces correspondientes se han creado los siguientes módulos *Python*:

- `app.py` contiene la propia aplicación `Dash` que es ejecutada.
- `lanzador.py` hace referencia a la página principal
- `interfaz.py` implementa la interfaz y funcionalidades para la lectura del Balance de Estado y su visualización
- `detalles_modelo.py` implementa la interfaz y las funcionalidades para visualizar y modificar los parámetros preconfigurados.

8.3.3. Diseño de las interfaces

La aplicación web se va a implementar como una única página principal, de carga dinámica y con un menú que determina las diferentes opciones. En esta etapa, dichas opciones son: pantalla inicial, visualizar el Balance de Estado y visualizar-modificar los parámetros preconfigurados.

Los cambios sobre la página se van a realizar sobre una parte de ella, denominada zona de intercambio, y sobre la que se muestra la información pedida, sin necesidad de recargar la página completamente. Los siguientes diagramas de navegación y de presentación[2] (*Figuras 8.1 y 8.2*) ilustra el modelo de representación de la aplicación.

8.3.4. Validación y pruebas

Para validar el correcto comportamiento del *Script* de clasificación se realizan dos pruebas de caja negra, recogidas en las *Tablas 8.3 y 8.4*.

[2] Véase [24]

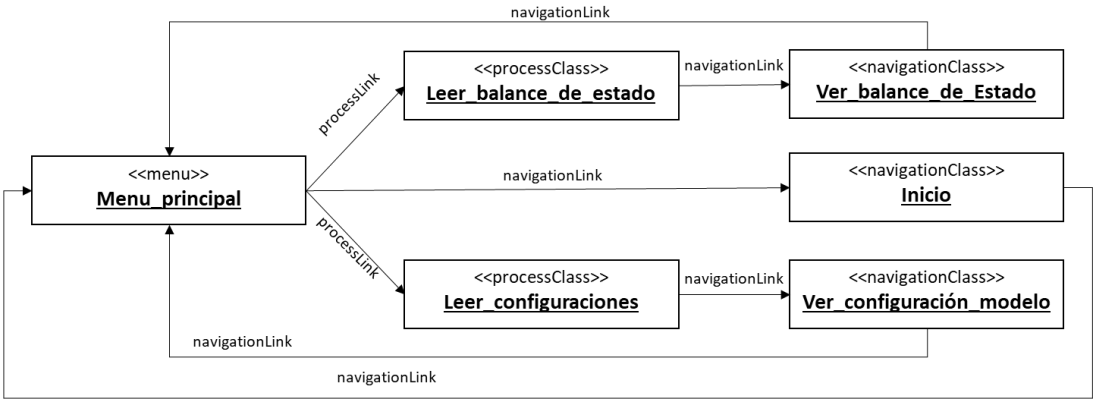


Figura 8.1: Diagrama de navegación para las interfaces de la etapa segunda.

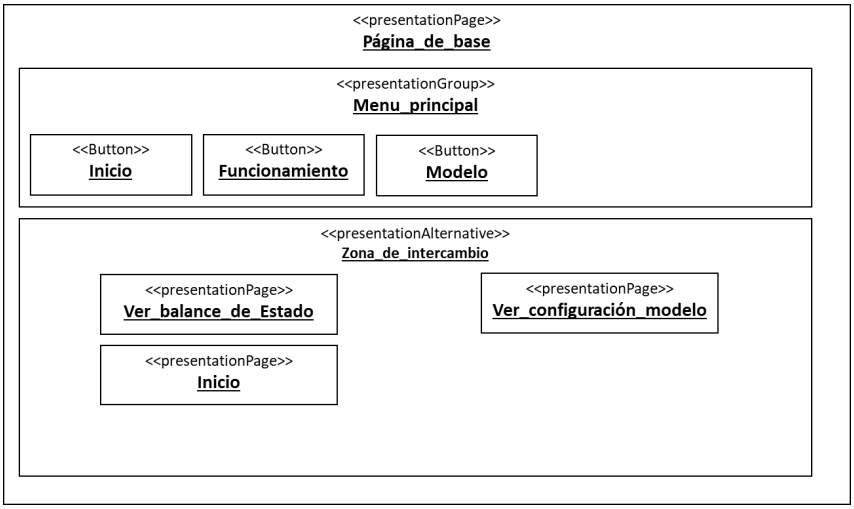


Figura 8.2: Diagrama de presentación para las interfaces desarrolladas en la segunda etapa

8.4. Tercera etapa

En la última fase se desarrollan todas aquellas funcionalidades que permiten la realización del estudio experimental, necesario para determinar los parámetros óptimos para la Configuración. Por tanto, se van a implementar los casos de uso **RU-05** y **RU-06**. Al igual que en la primera etapa, esta se desarrolla en 4 fases:

Fase 1: Construcción de las funcionalidades para evaluar los modelos. Se añaden al módulo `modulo_clasificacion`, que representa el Modelo de Clasificación, las funcionalidades que permiten calcular la distancia intra-clúster, varianza intra-clúster y representar el porcentaje de realizaciones de actividades presentes en cada clúster.

PCN-03	Generar un modelo con la Configuración introducida por el Usuario
Prerrequisitos	Existen la Configuración predefinida
Acciones	1. El Usuario solicita generar el modelo de clasificación según los parámetros propios, seleccionando la opción "Generar Modelo". 2. El Usuario modifica los parámetros de la Configuración que desee y selecciona la opción "Genera".
Resultado esperado	Se generan los modelos de clasificación para las operaciones y para los arranques, guardados como archivo `pickle` y denominados `modelo_arranque.pickle` y `modelo_operaciones.pickle` con las configuraciones específicas introducidas por el Usuario
Resultado obtenido	Se tiene el resultado esperado
Valoración de la prueba	Superada con éxito

Tabla 8.6: Prueba de caja negra sobre la generación de los modelos de clasificación con configuración propia.

Fase 2: Implementación del estudio experimental para la clasificación Se diseñan e implementan los métodos que, dados un conjunto de posibles configuraciones (rangos para los parámetros p y s descritos en la *Sección 5.3.1*), construyen los distintos modelos posibles y representan su valoración con la distancia intra-clúster como función objetivo.

Fase 3: Construcción de una interfaz para la clasificación Se desarrolla la interfaz correspondiente para solicitar al Usuario los rangos de configuraciones a estudiar y mostrar los resultados obtenido (*Fase 2*).

Fase 4:Interfaz para la tolerancia Se desarrolla la interfaz que visualiza los porcentajes de actividades en cada clúster y permite al Usuario realizar el estudio experimental sobre el valor de tolerancia (representado por α en la *Sección 5.3.1*).

8.4.1. Requisitos Funcionales

Los requisitos funcionales desarrollados en esta etapa se recogen en la *Tabla 8.8*.

8.4.2. Detalles de la implementación

En esta última etapa no se utiliza ningún módulo *Python* nuevo y que anteriormente no haya sido utilizado. Nuevamente, las interfaces se implementan con `Dash`.

PCN-04	Visualización del Balance de Situación
Prerrequisitos	Existen los Modelos de Clasificación para las operaciones y para los arranques
Acciones	El Usuario solicita visualizar el Balance de Situación de la máquina fresadora
Resultados esperado	El sistema muestra el Balance de Situación y lo actualiza cuando corresponde.
Resultados obtenido	Se tiene el resultado esperado
Valoración de la prueba	Superada con éxito

Tabla 8.7: Prueba de caja negra sobre la evaluación de las actividades

Id	Requisitos funcionales
RF-28	El sistema ofrece al Usuario (**ACT-01**) la opción de realizar el estudio experimental para la configuración de la clasificación.
RF-29	El sistema solicitará al Usuario (**ACT-01**) la información necesaria para realizar dicho estudio: fuente de datos, fecha inicial, duraciones máximas y mínimas, tipo de operación y tipo de material, rango de clústeres para p y s.
RF-30	El sistema tomará los datos de un archivo `pickle` siempre que se solicite esta opción como entrada de datos.
RF-31	El sistema tomará la información de dos archivos `csv` que solicitará siempre que se elija esta opción como fuente de datos
RF-32	El sistema construirá un Modelo de Clasificación para cada configuración en el rango establecido
RF-33	El sistema detectará los posibles problemas con la fuente de datos y los notifica.
RF-34	El sistema ofrecerá la opción de realizar un estudio experimental para el valor de tolerancia.
RF-35	El sistema solicitará al usuario el tipo de operación y tipo de material cuyo Modelo de Clasificación desea analizar.
RF-36	El sistema representará el porcentaje de elementos presentes en cada clúster del modelo respecto del total.
RF-37	El sistema ofrecerá la opción de introducir un valor candidato de tolerancia.
RF-38	El sistema representará el porcentaje de operaciones consideradas usuales y anómalas para el valor candidato de tolerancia dado.
RF-39	El sistema validará que el valor de tolerancia este entre 0 y 1 y lo notificará en caso contrario.

Tabla 8.8: Requisitos Funcionales de la Etapa 0

En cuanto a los módulos propios implementados, se han construido los siguientes:

- `estudio_procesado` recoge las funcionalidades implementadas en la *Fase 2* para generar

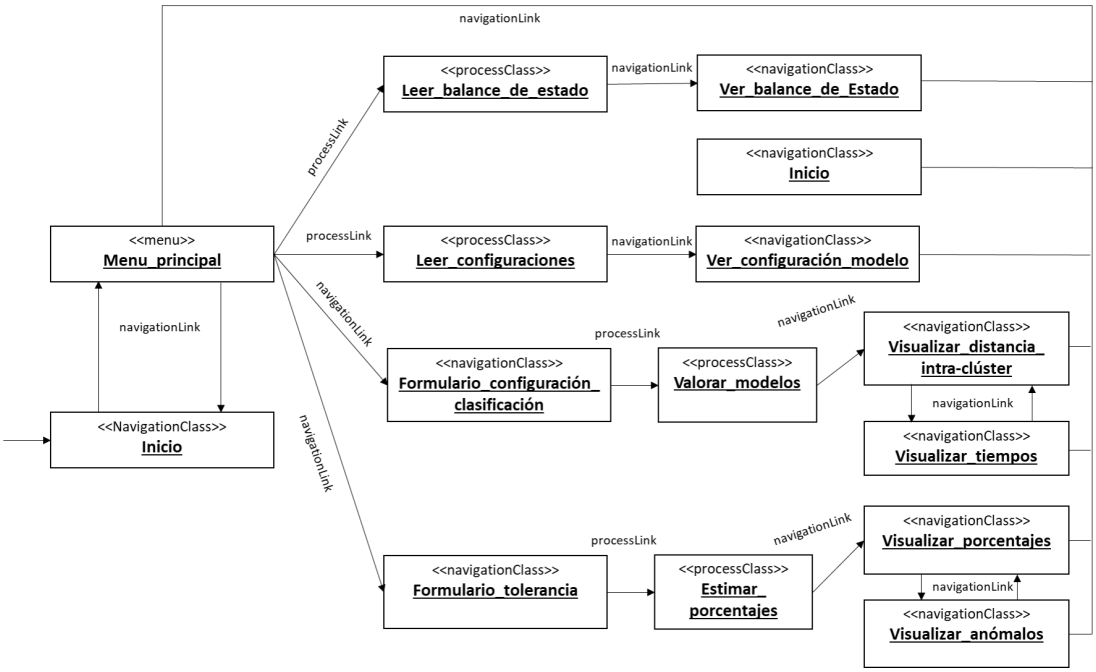


Figura 8.3: Diagrama de navegación de interfaces completo

los modelos según un rango de configuraciones y valorarlos.

- `estudio_configuracion_cluster` contiene la implementación de la interfaz para el estudio experimental de la configuración para la clasificación: toma de datos al Usuario y representación de la función objetivo.

- `estudio_tolerancia` contiene la interfaz para representar los porcentajes de cada clúster, así como, el conjunto de operaciones consideradas anómalas para cierto valor de tolerancia.

8.4.3. Diseño de las interfaces

Se complementan las interfaces diseñadas en la *Sección 8.3.3* para añadir las nuevas funcionalidades. Por tanto, los diagramas presentados en dicha sección son actualizados en las *Figuras 8.3 y 8.4*

8.4.4. Validación y pruebas

Para validar el correcto comportamiento de las funcionalidades implementadas en esta etapa, se realizan tres pruebas de caja negra, recogidas en las *Tablas 8.9, 8.10 y 8.11.*

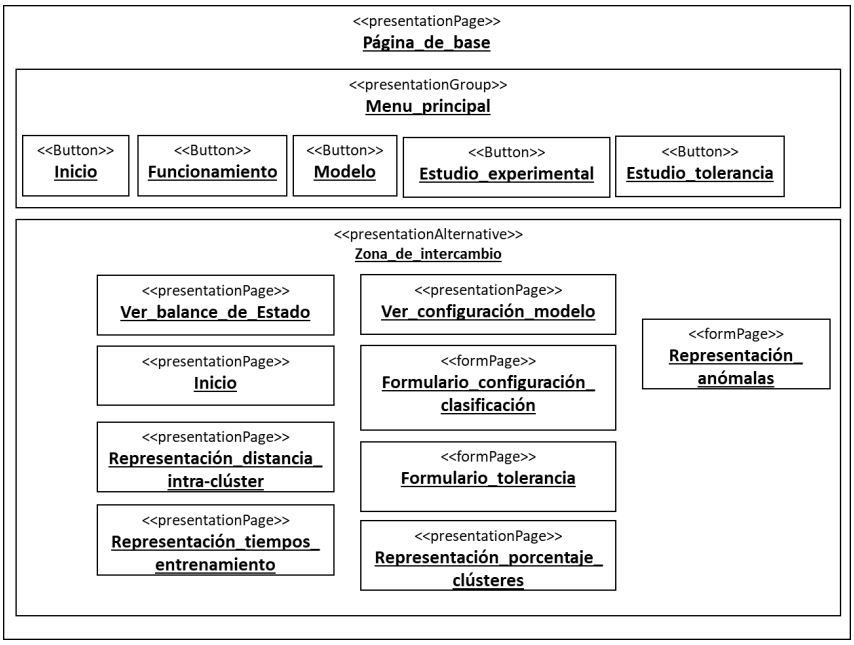


Figura 8.4: Diagrama de presentación de interfaces completo

PCN-05	Estudio experimental de las configuraciones para el modelo
Prerrequisitos	Hay actividades registradas en la base de datos.
Acciones	1. El Usuario solicita realizar el estudio experimental para la clasificación de la configuración seleccionando la opción "Estudio experimental". 2. El Usuario introduce toda la información necesaria: fuente de datos, fecha inicial, duraciones máximas y mínimas, tipo de operación y tipo de material, rango de clústeres para p y s (*Sección 5.3.1*).
Resultado esperado	▪ Se representa la función objetivo dada por la distancia intra-clúster respecto de los posibles valores de s y para cada valor de p. ▪ Se representan los tiempos requeridos para entrenar cada modelo.
Resultado obtenido	Se tiene el resultado esperado
Valoración de la prueba	Superada con éxito

Tabla 8.9: Prueba de caja negra sobre el estudio para la clasificación

PCN-6	Visualización del porcentaje por clústeres
Prerrequisitos	Existen los Modelos de Clasificación para las operaciones y para los arranques
Acciones	El Usuario solicita visualizar el porcentaje de operaciones en cada clúster para determinados tipos de *opermat*.
Resultados esperado	El sistema representa el porcentaje de actividades en cada clúster para cada uno de los modelos correspondientes con algún valor de *opermat* introducido.
Resultados obtenido	Se tiene el resultado esperado
Valoración de la prueba	Superada con éxito

Tabla 8.10: Prueba de caja negra sobre el estudio del valor de tolerancia I

PCN-7	Visualización del porcentaje de actividades consideradas anómalas para cierto valor de tolerancia
Prerrequisitos	Existen los Modelos de Clasificación para las operaciones y para los arranques
Acciones	1. El Usuario solicita visualizar el porcentaje de operaciones en cada clúster para determinados tipos de *opermat*. 2. El Usuario introduce el candidato a valor de tolerancia que se va a analizar.
Resultados esperado	El sistema representa el porcentaje de actividades usuales y anómalas para cada uno de los modelos correspondientes con algún valor de *opermat* introducido.
Resultados obtenido	Se tiene el resultado esperado
Valoración de la prueba	Superada con éxito

Tabla 8.11: Prueba de caja negra sobre el estudio del valor de tolerancia II

Capítulo 9

Estudio experimental para la Configuración del Modelo

En la *Sección 5.3* se describió la arquitectura y funcionamiento del Modelo de Clasificación que se utiliza. Además, se determinó que este modelo dependía de tres parámetros predefinidos recogidos en la *Sección 5.3.1*. En este capítulo, mediante un estudio experimental en el que se valoran diferentes configuraciones tras construir sus respectivos modelos, se busca determinar los parámetros óptimos. Además, dicho estudio permite marcar la importancia de estos parámetros sobre la bondad de los resultados obtenidos.

9.1. La configuración para la clasificación

En esta sección se busca determinar la mejor *configuración para la clasificación* dada por los parámetros p y s (véase 5.3.1). En esta sección se muestra con todo detalle el estudio y los resultados para los arranques de la máquina fresadora. En el *Apéndice F* se incluyen los resultados gráficos para el resto de las actividades, siendo el estudio análogo.

La técnica aplicada es el **método del codo** con la distancia intra-clúster como función objetivo.

Sobre el algoritmo de k-medias tradicional, el método del codo supone que existe una función objetivo g real que traduce algunas variables o características, dependientes de los parámetros que determinan el algoritmo de clasificación (en k-medias el número de clústeres a construir), y los traduce en un valor real. Representando de forma continua la función objetivo respecto de los parámetros, este método afirma que la elección óptima se da, gráficamente, en el cambio de un decrecimiento brusco a uno moderado de la función objetivo. La *Figura 9.1* ilustra un ejemplo de aplicación del método del codo sobre una función objetivo.

En el caso de estudio, la función objetivo es la distancia intra-clúster (*Definición D.1*) inducida por la DTW como medida de similitud.

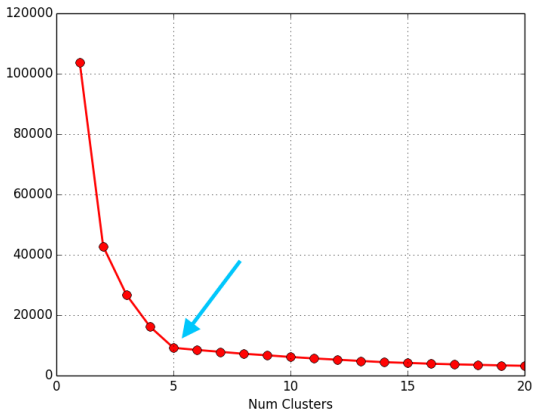


Figura 9.1: Ejemplo de aplicación del método del codo. La solución es de 5 clústeres. Fuente: jarroba

Dado que se trata de un modelo híbrido, en una primera aproximación se analiza la mejor elección de clústeres para dividir la duración (parámetro p). Considerando como función objetivo la distancia intra-clúster, con el módulo de las diferencias entre duraciones induciendo la medida de comparación, se obtiene la representación recogida en la *Figura 9.2*.

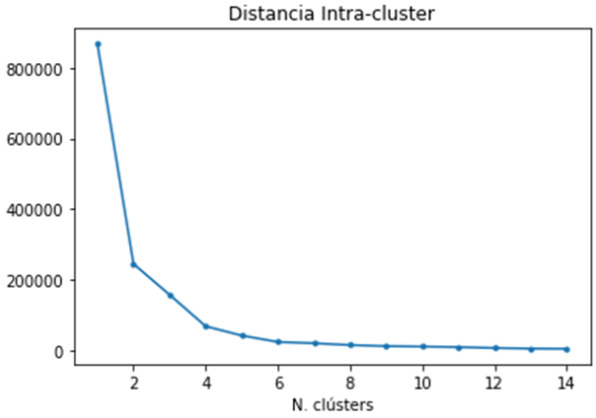


Figura 9.2: Representación de la distancia intra-clúster para diferentes configuraciones de número de clústeres para las duraciones, para el tipo de operación de contorneado sobre acero.

Los cambios en el decrecimiento se producen para 2 y 4 clústeres. Si solo se considerase dicho parámetro, el método del codo elegiría como mejor configuración una división de 4 clústeres. Sin embargo, no se pueden considerar los parámetros p y s independientes. Así, la conclusión que se obtiene del estudio anterior es para el parámetro p basta con estudiar los valores 2, 3, 4 ó 5.

Para estudiar la configuración para la clasificación, se considera la distancia intra-clúster bajo la DTW para los clústeres finales como función objetivo. Para cada posible valor de p, se representan los posibles valores para s frente a la función objetivo. Esto se recoge en la *Figura 9.3*

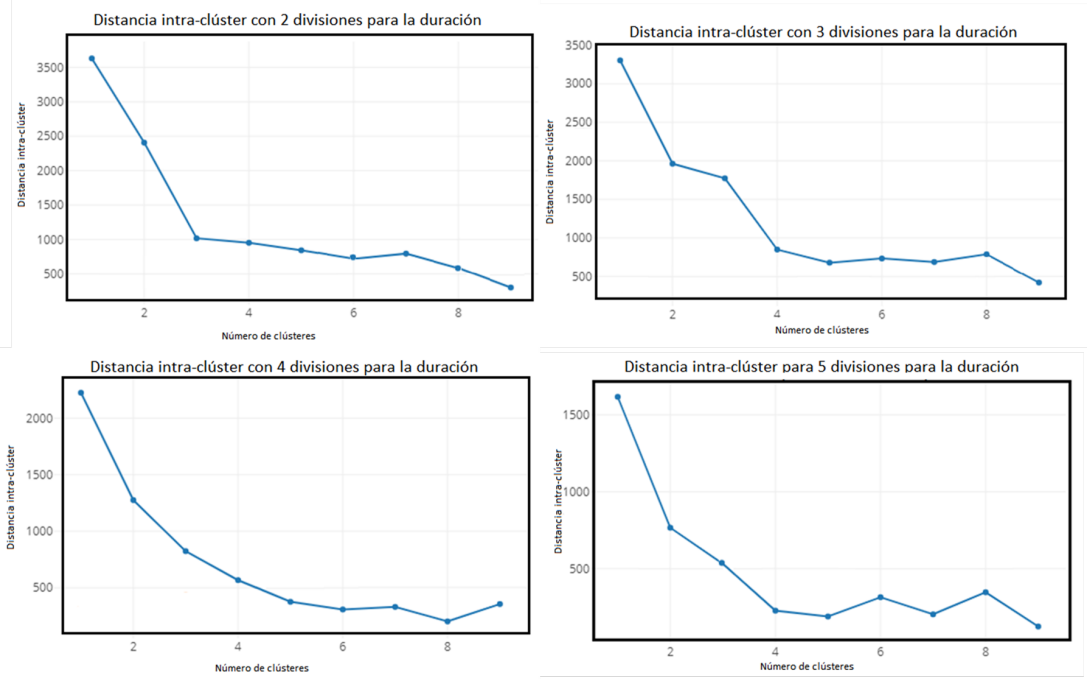


Figura 9.3: Representación de la distancia intra-clúster como función objetivo, resultado del estudio de la configuración para la clasificación

Si se denota por el par (p, s) la configuración para la clasificación, entonces aplicando el método del codo sobre la representación de la función objetivo dadas en la *Figura 9.3*, se tienen que las mejores configuraciones son $(2, 3)$, $(3, 4)$, $(4, 4)$ y $(5, 4)$.

La elección final se ha basado en la experiencia con los arranques que se disponía dentro del proyecto IDIGIT4L, que consideraba excesivas 12 divisiones para los arranques. Además, el valor $p = 2$ es consistente con el método del codo aplicado a las duraciones, siendo poco notable la diferencia de los valores de la distancia intra-clúster entre las configuraciones anteriores. Por todo ello, se elige $(2, 3)$ como la configuración para la clasificación de los arranques.

De forma análoga se realiza el estudio para el resto de las actividades (*opermats* de 23, 33, 53 y 63), cuyas representaciones de la distancia intra-clúster se recogen en el *Apéndice F*. En este caso, con objetivo de mantener un criterio homogéneo para las operaciones, se busca que la configuración elegida sea única y válida para los cuatro modelos. Por ello, se elige $(2, 5)$ como configuración para la clasificación de las operaciones.

9.2. Resultados de la clasificación

Una vez construido el modelo de clasificación y entrenado con un conjunto de datos, se procede a su estudio con el objetivo de discriminar los clústeres usuales o anómalos. Se supone que todas las series temporales clasificadas en el mismo clúster comparten un modelo (proceso estocástico) común.

Por tanto, si un clúster tiene asignadas una gran cantidad de realizaciones de una misma actividad, dicho clúster es considerado usual, y, las actividades asignadas a dicho clúster tienen un comportamiento esperado, sin requerir correcciones. En caso contrario, si un clúster tiene asignadas pocas realizaciones, entonces este es anómalo y las realizaciones asignadas o la propia máquina pueden requerir de una revisión.

9.2.1. El valor de tolerancia

Para cuantificar si un clúster tiene asociadas muchas o pocas realizaciones, se estudia el tercer parámetro del Modelo de Clasificación: el valor de tolerancia (*Sección 5.3.1*), denotado por α. Así, un clúster (respectivamente una actividad asociada a dicho clúster) es usual si el porcentaje de realizaciones asignadas respecto del total es superior a α, y anómalo en caso contrario.

Para cada tipo de actividad para los cuales se construyó un Modelo de Clasificación (operaciones de Planeado, Contorneado, Fresado y Taladrado sobre el acero -*opermat* de 23, 33, 53 y 63-, y los arranques de la máquina -*opermat de* −10-) se analizan sus respectivos modelos, construidos los parámetros (p, s) determinados en la sección anterior (*9.1*), y se determina el valor de tolerancia.

Al igual que en la sección anterior, se desea mantener una cierta homogeneidad en los modelos de las operaciones. Por ello, se va a determinar un único valor de tolerancia para todos los tipos de operación y otro valor de tolerancia para los arranques.

En primer lugar, se representan (*Figura 9.4*) los porcentajes de actividades asignados a cada clúster para cada uno de los Modelos de Clasificación construidos. De esta representación, se deduce que el valor de α debe estar entre 0,05 y 0,15 para las operaciones y entre 0,10 y 0,20 para los arranques.

La técnica experimental para determinar con mayor exactitud el valor de tolerancia α se basa en analizar el número de realizaciones consideradas anómalas y usuales para distintos valores candidato de tolerancia, y elegir el valor anterior a aquel que "dispara" el número de operaciones consideradas anómalas. Además, gracias a la experiencia con la máquina fresadora, se ha podido estimar que entre el 15 % y el 30 % de las actividades tienen un comportamiento anómalo.

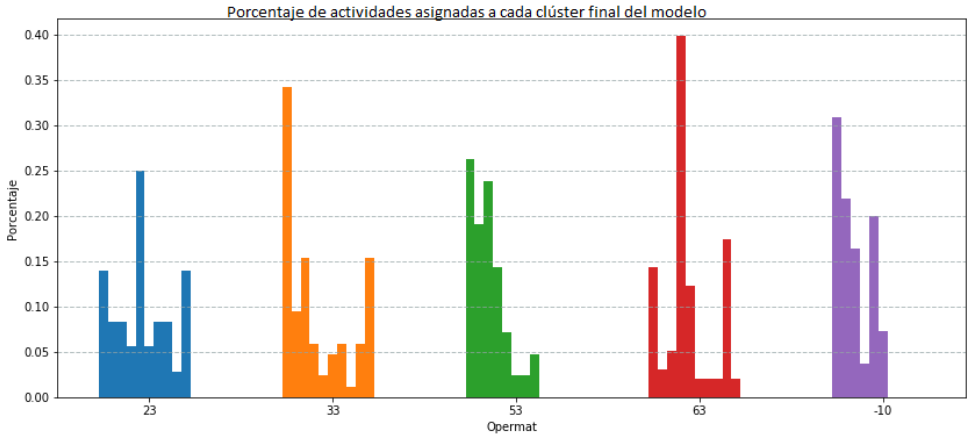


Figura 9.4: Representación del porcentaje de actividades asignados a cada clúster final del modelo para cada *opermat*.

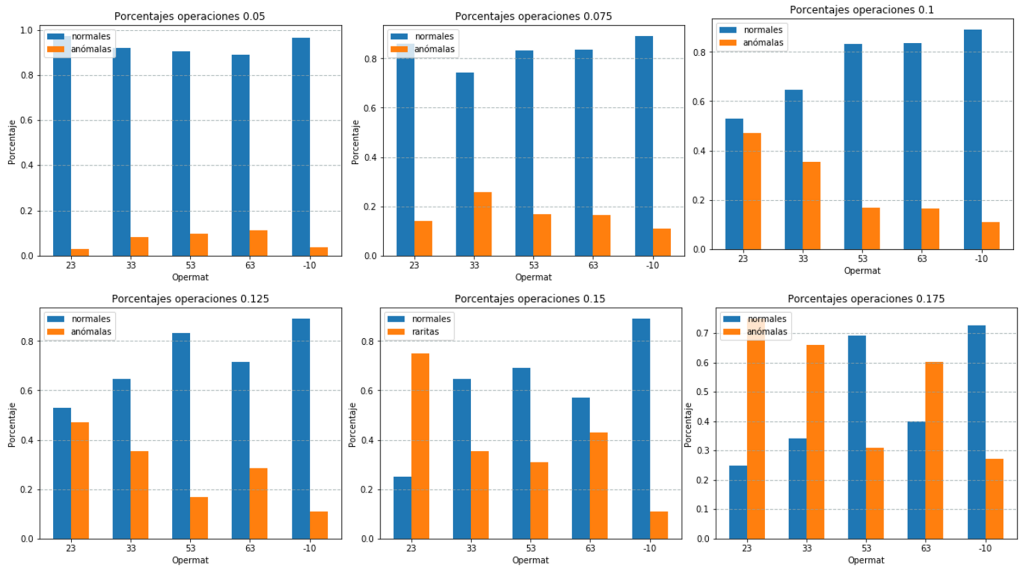


Figura 9.5: Representación del porcentaje de actividades consideradas normales y anómalas para diferentes valores de tolerancia discriminando por el *opermat*

En la *Figura 9.5* se recoge dicho estudio, y se determina como valor de tolerancia para los arranques 0,15 y para las operaciones de 0,10 (aunque 0,125 también podría ser considerado válido). Estos valores de tolerancia considerados discriminan como actividades anómalas entorno al 20 – 25 % de las operaciones y 10 – 15 % de los arranques. Además, si se considera como valor de tolerancia 0,15 para las operaciones y 0,17 para los arranques, el porcentaje de las actividades consideradas anómalas se sitúa por encima del 40 %.

9.3. Evaluación del modelo

En esta sección se busca valorar la consistencia de los Modelos de Clasificación construidos. El propio modelo indica qué actividades (operaciones o arranques) del conjunto de entrenamiento son poco frecuentes. Además, dada una nueva actividad, esta se puede evaluar mediante el modelo correspondiente (*Sección 5.3.2*).

Siguiendo la técnica expuesta en dicha sección, se evalúan los datos desde el 1 de mayo-2019 al 1 de septiembre-2019[1] con el respectivo modelo construido. Bajo la hipótesis de que la probabilidad de que una actividad sea anómala es constante en el tiempo, se evalúa si el porcentaje de operaciones usuales y anómalas detectado con el conjunto de entrenamiento se mantiene con el conjunto de evaluación.

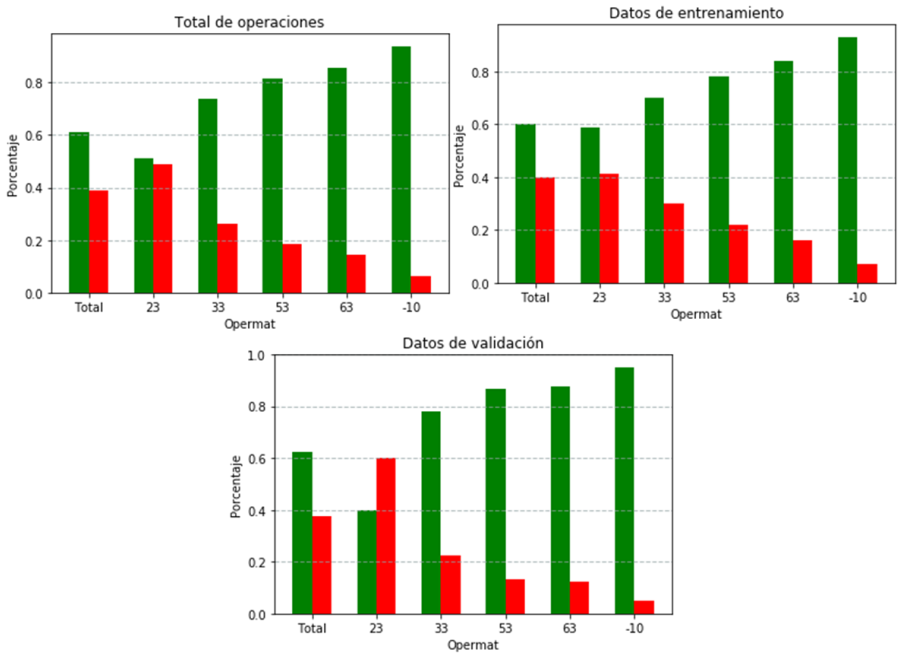


Figura 9.6: Representación de los resultados de evaluar las actividades realizadas con sus correspondientes modelos generados, para la totalidad de los datos, los datos de entrenamiento (septiembre-19 en adelante) y los datos de prueba (de mayo-19 a septiembre-19). En verde los clasificados como usuales y en rojo anómalos

[1]Se disponen de datos desde mayo-2019 hasta mayo-2020 para los arranques o marzo-2020 para las operaciones. Los respectivos modelos se entrenan con los datos disponibles a partir de septiembre-2019 (inclusive) y se valida sobre los datos recogidos entre mayo-2019 hasta el día 1 septiembre-2019. Al primer subconjunto se le denomina datos de entrenamiento y al segundo conjunto se le denomina datos test. La proporción entre ambos conjuntos es respectivamente de, aproximadamente, 2/3 y 1/3, respecto del total.

En la *Figura 9.6* se ilustran los resultados. Como consecuencia, se tiene que los modelos son consistentes en el tiempo (salvo para la operación de planeado sobre acero *opermat* 23), suponiendo que la frecuencia en la que se producen las anomalías se mantiene en el tiempo. En el caso de la operación con código 23, esta divergencia puede justificarse en que el conjunto de entrenamiento para este tipo de actividad presentaba menos realizaciones que en el resto de las actividades.

Por tanto, se puede aceptar la hipótesis siguiente: "cuando el conjunto de entrenamiento tiene las suficientes realizaciones de la actividad estudiada, el Modelo de Clasificación construido es consistente en el tiempo".

9.4. Ventajas del modelo híbrido

El algoritmo de k-medias bajo la distancia DTW presenta el inconveniente de tener complejidad polinómica de orden 5, $\mathcal{O}(n^5)$, dado que el cálculo de la distancia DTW entre dos series tiene complejidad cuadrática. Por contra, el problema de clasificación por k-medias tradicional para conjuntos de datos unidimensionales es un problema de complejidad, a lo sumo, cuadrática.

Como consecuencia, al dividir el conjunto total de las series temporales en subconjuntos más pequeños utilizando el algoritmo de k-medias tradicional, y aplicar sobre dichos subconjuntos el algoritmo de k-medias bajo la DTW se obtiene una notable mejora en la eficiencia del modelo. En la *Figura 9.7* se muestra el tiempo requerido para diferentes configuraciones, de la operación de contorneado sobre acero, siendo destacable la diferencia del tiempo consumido al clasificar sin dividir por la duración respecto de realizar una clasificación dos divisiones.

Por otro lado, esta arquitectura de modelo híbrido minimiza el inconveniente de la DTW: deformaciones temporales excesivas. Pues, fijando el tamaño de los centroides como el mínimo de la longitud de las series a clasificar, y dada la alta variabilidad de los tamaños de series considerados (de seis minutos a una hora), es posible y usual que la longitud de la serie a clasificar pueda sextuplicar la longitud del centroide. Agrupando por la duración se consigue reducir la máxima diferencia entre la longitud de las series a clasificar bajo la DTW.

Para concluir este capítulo, se debe comentar la importancia de los parámetros del modelo. Cuando se produce una anomalía, lo que se está produciendo en una desviación, en un fragmento de la serie temporal, respecto al modelo de la actividad considerado como usual. Por tanto, dos anomalías diferentes suelen tener una distancia mayor entre sí que respecto a una acción usual de la que se desvían (en general, los clústeres anómalos tienen menos de 5 % de los elementos). Por ello, si el número de clústeres no es lo suficientemente grande, habrá anomalías que se asignen al clúster mayoritario (usual).

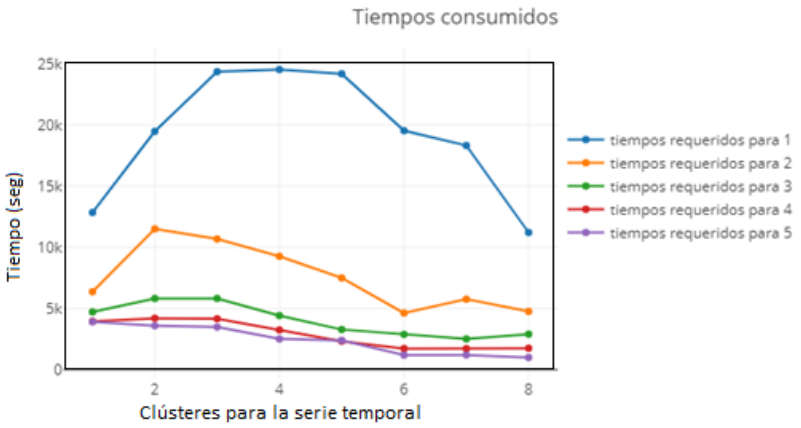


Figura 9.7: Representación de los tiempos requeridos en la construcción de modelos con diferentes configuraciones de la operación de contorneado sobre el acero. Cada representación se corresponde con un valor fijo de clústeres para la duración para cada número de clústeres para la clasificación bajo DTW.

Por otro lado, el valor de tolerancia α puede resultar engañoso y los resultados presentan una gran dependencia. Por ejemplo, un valor de α de 0,1 marca como anómalas el $23,67\,\%$ de las operaciones, de 0,2 detecta el $57{,}66\,\%$ de las operaciones, y un valor de 0,3 el $70,86\,\%$. Por contra, un valor de α de $0,05$ marca como anómalas un insuficiente $9\,\%$ de las operaciones.

Capítulo 10

Conclusiones y líneas de trabajo futuras

10.1. Conclusiones y consecución de objetivos

Este trabajo ha sido enfocado a la resolución del *problema de detección de anomalías sobre un conjunto de datos estructurados como series temporales*, derivado del problema de detectar las actividades anómalas realizadas en una máquina fresadora CNC. El fin último es la construcción de un sistema semiautónomo, capaz de detectar las actividades anómalas y notificarlas al operador de la máquina fresadora.

Para el problema de diferenciar las actividades entre usuales y anómalas, se ha propuesto *un modelo de clasificación no supervisada o clustering de enfoque híbrido*. Este, realiza una doble clasificación, atendiendo en primer lugar a la duración de la serie, y seguidamente, usando la propia serie temporal como dato (*Capítulo 5*).

Para esta solución, se ha adaptado el algoritmo de k-medias (*k-means*) a su uso sobre conjuntos de series temporales de longitud no definida. Esto se ha logrado utilizando la distancia *Dynamic Time Warping* como medida de similitud y la media de Fréchet según el algoritmo *DTW Barycenter Average* como función promedio (*Capítulo 4*).

El Modelo de Clasificación propuesto ha hecho posible el desarrollo de un sistema, denominado *Sistema de Detección de Anomalías*, capaz de construir los modelos y valorar las actividades realizadas. Además, este sistema ha sido complementado, permitiendo realizar los estudios sobre la configuración adecuada para cada modelo.

En este punto, para concluir el proyecto es necesario recuperar los objetivos definidos al comienzo (*Sección 1.2*), y valorar si se han alcanzado.

En primer lugar, del estudio de la máquina fresadora se determinó que las actividades (operaciones y arranques) debían ser diferenciadas según el tipo de operación y el tipo de material, y representadas mediante la serie temporal resultado de las mediciones del sensor de corriente de fase 1 (**Obj-01**).

Seguidamente, la información almacenada en la base de datos ha sido estructurada (**Obj-02.1**), y se ha eliminado aquellas actividades problemáticas o mal etiquetadas (**Obj-02.2**). Hecho esto, teniendo en cuenta la variabilidad de las series, con las fortalezas y debilidades de la DTW, se ha propuesto (**Obj-02.3**) e implementado (**Obj-02.4**) una arquitectura de Modelo de Clasificación, adaptada a las peculiaridades del caso de estudio (*Capítulo 5*).

Sin embargo, la arquitectura propuesta queda caracterizada por una serie de parámetros, *a priori*, desconocidos y de gran influencia en los resultados. Estas variables son determinadas mediante un estudio experimental (*Capítulo 9*), basado en el método del codo con función objetivo la distancia intra-clúster inducida por la DTW (**Obj-02.5**).

Además, construidos los modelos de clasificación, estos son consistentes en el tiempo (**Obj-02.6**), bajo la hipótesis de que la probabilidad de obtener una actividad anómala es constante, y siempre que el conjunto de entrenamiento tenga suficientes realizaciones del tipo de actividad correspondiente.

Con el Modelo de Clasificación como eje principal, se ha diseñado e implementado el *Sistema de Detección de Anomalías*, integrado dentro del proyecto IDIGIT4L. Para ello, se ha optado por un desarrollo modular de un sistema con dos componentes, donde cada módulo cumple una función específica (*Capítulo 6*).

De esta forma, se ha implementado un módulo capaz de conectar con la Base de Datos, externa al proyecto y vehículo de conexión con el resto de las componentes del proyecto IDIGIT4L, para leer la información de los sensores (**Obj-03.1**). Para procesar esta información, se ha elaborado un *script* capaz de generar los Modelos de Clasificación (**Obj-03.2**); y , otro *script*, que apoyándose en los modelos valora "en vivo" las actividades realizadas (**Obj-03.3**) (*Capítulos 7 y 8*).

10.2. Aprendizaje personal

En esta sección, presento brevemente unas notas sobre mi experiencia personal con este trabajo y las aportaciones que me ha supuesto su realización.

Llevar a cabo este proyecto me ha supuesto una aplicación de todos los conocimientos adquiridos durante la carrera, teniendo especial importancia los conocimientos relacionados con las asignaturas que tratan la *Ingeniería de Software*. Pues, aunque por la naturaleza intrínseca del proyecto, el sistema presentado y su desarrollo se alejan en cierta medida del análisis, diseño

e implementación tradicionales, con un ciclo de vida evolutivo y marcado por factores externos, el presente trabajo me ha permitido aplicar todos mis conocimientos adquiridos para diseñar y planificar el Sistema de Detección de Anomalías.

Por otro lado, dado que su aplicación se ha realizado sobre un caso real, lejos de los ejemplos académicos y dentro de las prácticas en la Fundación CIDAUT, me ha permitido enfrentarme a un problema real, con restricciones externas e integrado dentro de un proyecto de mayor envergadura.

Además, fuera del ámbito teórico y del desarrollo de esta memoria, este trabajo me ha permitido conocer el funcionamiento e integrarme dentro de un equipo de desarrollo, donde cada compañero realizaba distintas funciones. Es más, este proyecto no habría sido posible sin la ayuda, la experiencia y los comentarios de mis compañeros, y he podido trabajar como un miembro más del equipo dentro del proyecto IDIGIT4L de la Fundación CIDAUT. Pues ha sido su experiencia la que me ha permitido entender y comprender el funcionamiento de la máquina fresadora, así como decidir entre diferentes opciones y alternativas (como para la configuración del modelo). Asimismo, han sido ellos quienes se han encargado de sensorizar la máquina fresadora, almacenar la información recogida en la base de datos y con quienes se ha conectado el *Script* de Clasificación al sistema de visualización desarrollado en *Grafana*.

10.3. Líneas de trabajo futuras

Las líneas de trabajo futuras para completar este proyecto se centran en tres frentes.

- Integración completa del Sistema de Detección de Anomalías dentro del proyecto IDIGIT4L, con objeto de que forme parte de la lógica interna. Para su integración, el *Script* de Clasificación debería ser capaz de ejecutarse dentro de un sistema de sensorización compacto, donde se encargaría de la construcción de los Modelos de Clasificación y la evaluación de las actividades registradas.

- Por otro lado, el segundo módulo del Sistema de Detección de Anomalías requiere de un servidor para su correcto funcionamiento. Así, se propone la integración del sistema completo en un entorno de producción funcional, donde el acceso a la aplicación web se pueda realizar desde cualquier dispositivo con conexión a Internet.

- El Sistema de Detección de Anomalías se puede mejorar, y evolucionar en sus valoraciones. Para ello, con la intervención del operario de la máquina fresadora, se deberían etiquetar aquellas actividades consideradas anómalas por el Modelo de Clasificación. Con esto, se identifican las posibles causas asociadas a cada clúster anómalo. Además, dichas causas tendrán asociadas a un conjunto de acciones que el operario debe realizar para reparar o

minimizar sus consecuencias, tanto sobre la propia pieza como sobre la máquina fresadora.

- El modelo presentado en el *Sección 5.3* puede ser mejorado. En primer lugar, para la clasificación por el algoritmo de k-medias tradicional se pueden utilizar otros atributos de la serie temporal (como la media o varianza, bajo el supuesto de que las series de las operaciones son estacionarias). Por otro lado, la clasificación por series temporales puede mejorarse utilizando más información, como las mediciones de los acelerómetros o de la temperatura.

- Por último, el sistema puede ser mejorado si se atiende a la estacionalidad de las series temporales (*Sección 5.2*). En este caso, se propone asumir la hipótesis de que las series que representan las operaciones tienen un modelo estacionario, y por tanto, cada medición o segmento pequeño de la propia serie puede ser evaluado sin necesidad de esperar a que finalice la actividad.
  Por ejemplo, siguiendo esta idea, se puede establecer unos valores de máximo y mínimo para cada medición, notificando cuando alguna de las mediciones se salga de dicho intervalo.

Apéndice A

Contenido entregado y manual de uso

A.1. Contenido entregado

En el repositorio https://uvaes-my.sharepoint.com/:f:/g/personal/subdireccion_euisg_uva_es/EmKtZU_j1KVDmI-zes8tEVIBc8uVP-88zmpF0-mm26WJsAi facilitado por la Universidad de Valladolid se adjunta la presente memoria y el *Sistema de Detección de anomalías*. El contenido de este último se encuentra comprimido en el archivo `SistemaDeteccionAnomalias.zip`. En el fichero comprimido se encuentran los siguientes archivos y directorios:

`configuracion_general.` Directorio que contiene el archivo *Python* que determinan todos los parámetros de configuración del SDA, denominado `configuracion.py`. También contiene el archivo `diccionario.py` que contiene las relaciones entre los identificadores y su nombre de las operaciones y los materiales.

`modulo_conexionBBDD.` Directorio que contiene el *script* que permite la conexión e intercambio de información con la Base de Datos (`conectarBBDD.py`) y el fichero con la configuración de la Base de Datos y las consultas (`configBBDD`).

`modulo_procesado.` . Este directorio contiene los *script* para el procesado de la información en bruto (`FuncionesPropias.py` y `MetodosProcesado.py`) y la definición de clases para el estructurado (`ClasesActividades.py`)

`modulo_clasificación.` Esta carpeta contiene un único script que define e implementa el Modelo de Clasificación (`ModuloClasificacion`).

`Interfaces.` Este directorio contiene las interfaces diseñadas para el sistema principal (`interfaz.py` y `modelo_info.py`).

`estudio_experimental.` Contiene los *scripts* e interfaces que permiten realizar el estudio para la configuración: `estuio_configuracion_cluster.py`, `estudio_procesado.py` y `estudio_valor_tolerancia.py`.

`Registros.` Contiene dos archivos `csv` donde se anotan las actividades evaluadas y las actividades anómalas.

`static`. Carpeta creada para la aplicación Dash, permite al servidor `flask` servir al cliente imágenes y reglas de estilo `css`.

`main.py.` *Script* de evaluación en vivo.

`creador_del_modelo` *Script* para generar el modelo a partir de los parámetros preconfigurados recogidos en `configuracion.py`.

`app.py y lanzador.py` *Scripts* claves para la ejecución del SDA completo. El primero contiene los detalles para montar el servidor y ejecutar la aplicación. El segundo es el programa principal con la interfaz inicial y el gestor de interfaces.

`estados.txt` Representa el Balance de Situación de la máquina fresadora.

`modelo_arranque_loosen.pickle y modelo_aoperaciones_loosen.pickle.` Modelo de Clasificación para el arranque y las operaciones, almacenado en archivo binario.

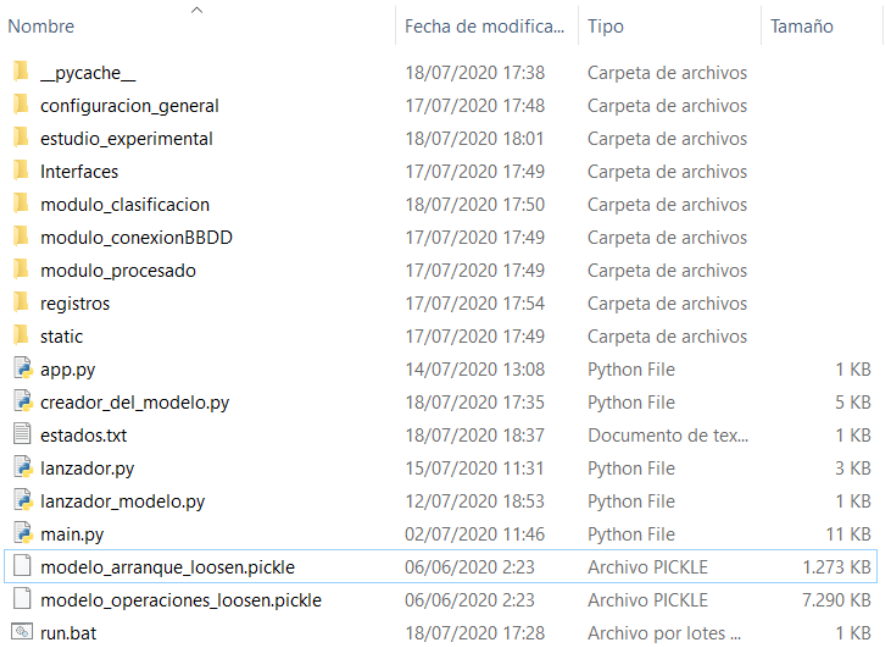


Figura A.1: Documentos incluidos en `SistemaDeteccionAnomalias.rar`

A.2. Manuales

En esta sección se incluyen los manuales de usuario para el Sistema de Detección de Anomalías, tanto para las interacciones del *Software Huésped* (el encargado de ejecutar) con el *Script* principal como para las interacciones del usuario mediante el módulo secundario.

A.2.1. Requisitos para poder utilizar el sistema

En esta sección se recogen los requisitos Software y Hardware necesarios para poder ejecutar el Sistema de Detección de Anomalías.

En primer lugar, cabe destacar que el sistema debe ser ejecutado en un dispositivo que tenga conexión con la Base de Datos utilizada. Por otro lado, dado que se ejecuta en un entorno *Python* sin recurrir a comandos u ordenes propios de algún sistema operativo, este es indiferente.

Sin embargo, el entorno *Python* para su ejecución debe tener instaladas los módulos *Python* descritos en la *Sección 2.3*.

En cuanto a memoria, esta varía enormemente entre las acciones que se realizan. Esta diferencia viene dada por la información recibida previo a su procesamiento por el *módulo de procesamiento y estructurado*. Así, un año de información sin procesar se estima entre 2,5 y 3 GB, mientras que esa misma información procesada se estima entre 250 y 500 MB. Por otro lado, los modelos de clasificación no requieren de demasiada memoria en comparación con las actividades, no superando los 50 MB. Esto se debe a que los modelos registran las actividades como `Dato_generico`.

En consecuencia, un sistema que no genere modelos y simplemente evalúe podría funcionar con 1 GB de memoria, mientras que si se requiere la construcción de modelos o el procesamiento de grandes cantidades de información se requerirían no menos de 3 GB de memoria. A esto habría que sumar la memoria requerida por el sistema operativo y por el propio intérprete de Python.

A.2.2. Manual de ejecución del Sistema de Detección de Anomalías

En esta sección se describen los pasos necesarios para la ejecución del *Script* de Clasificación de forma independiente y del sistema completo.

Comenzando por el *Script* de Clasificación de forma independiente, la configuración se puede modificar en el archivo `configuracion.py` dentro del directorio `configuracion_general`.

Para generar los modelos de clasificación se debe ejecutar el archivo `lanzador_modelo.py`. Así, suponiendo como directorio principal (actual) la carpeta `SDA` se tiene que ejecutar:

```
1 python lanzador_modelo.py
```

Por otro lado, para la evaluación de las actividades en vivo se debe ejecutar el *script* `main.py`.

```
1 python main.py
```

En cuanto al Sistema completo, basta ejecutar el archivo `lanzador.py`. Suponiendo que el directorio actual es `SDA` entonces:

```
1 python lanzador.py
```

En Windows, el Sistema de Detección de Anomalías completo se puede ejecutar utilizando el archivo `run.bat`, teniendo en cuenta que el *PATH* de *Python* debe estar añadido al intérprete de comandos del sistema.

A.2.3. Manual de uso de la aplicación

En esta sección se describen las opciones que ofrece el Sistema de Detección de Anomalías, y se ilustran las interacciones posibles con dicha aplicación.

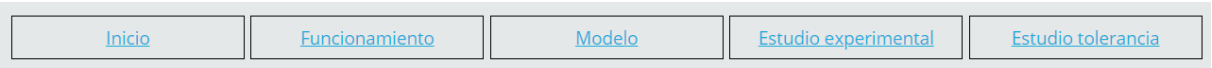


Figura A.2: Opciones ofrecidas por la aplicación

- **Inicio**: página inicial
- **Funcionamiento**: permite visualizar el Balance de Estado de la máquina (resultados de la evaluación y estado).
- **Modelo**: permite generar los modelos de clasificación con los parámetros propios introducidos por el usuario.
- **Estudio experimental**: permite realizar el estudio experimental, que busca determinar la mejor configuración para la clasificación (p y s en la *Sección 5.3.1*).
- **Estudio tolerancia**: permite realizar el estudio experimental sobre el valor de tolerancia para el Modelo de Clasificación construido.

Tras ejecutar el Sistema de Detección de Anomalías completo, y una breve espera, en la que se prepara el servidor, se puede acceder desde el navegador preferido en la dirección `127.0.0.1:8083`. Para ilustrar este manual, se ha usado *Google Chrome* como navegador web.

Al ingresar la dirección en el navegador nos dirige a la página de bienvenida (*Figura A.3*).

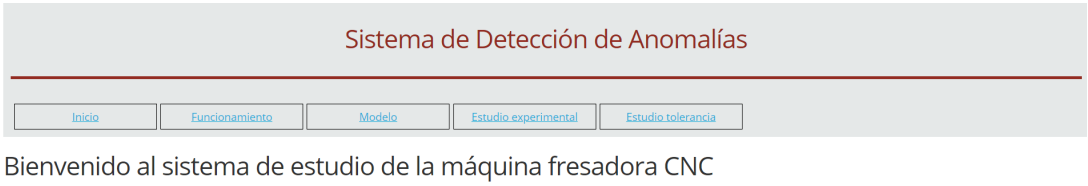


Figura A.3: Página de Bienvenida del SDA

Si se elige la opción de "Funcionamiento" nos muestra el Balance de Situación de la máquina fresadora, con el estado actual y el resultado de la última operación evaluada (*Figura A.4*). Esta página solo muestra información actualizada del Balance de Situación y no permite otro tipo de interacción con el usuario.

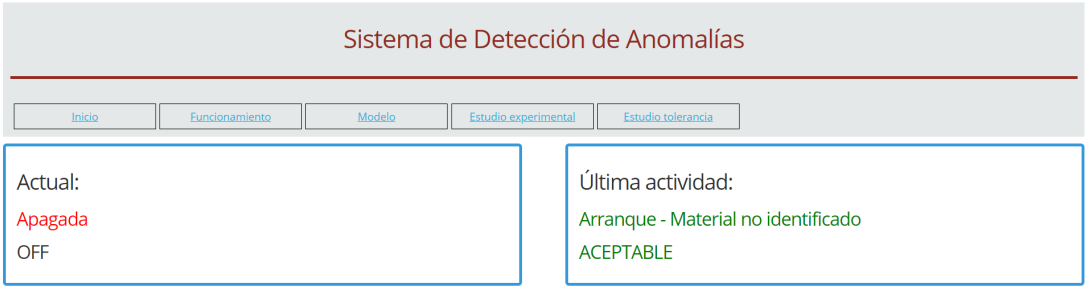


Figura A.4: Página de "Funcionamiento" del SDA

La opción "Modelo" muestra los parámetros de configuración, y permite su edición para generar un Modelo de Clasificación con una configuración determinada por el usuario. En esta pestaña (*Figura A.5*) se muestra la información de la Configuración (determinada por el archivo `configuracion.py`) y se permite su edición. Cuando se han realizado todas las modificaciones deseadas, el modelo se genera utilizando el botón "Generar nuevo modelo". Este procedimiento suele requerir de cierto tiempo, dependiendo de la fecha de inicio elegida. Previó a la generación del modelo, el sistema verifica que los campos introducidos sean correctos (en cuanto a formato y dominio de valores), y, en caso contrario, lo notifica al usuario (*Figura A.6*).

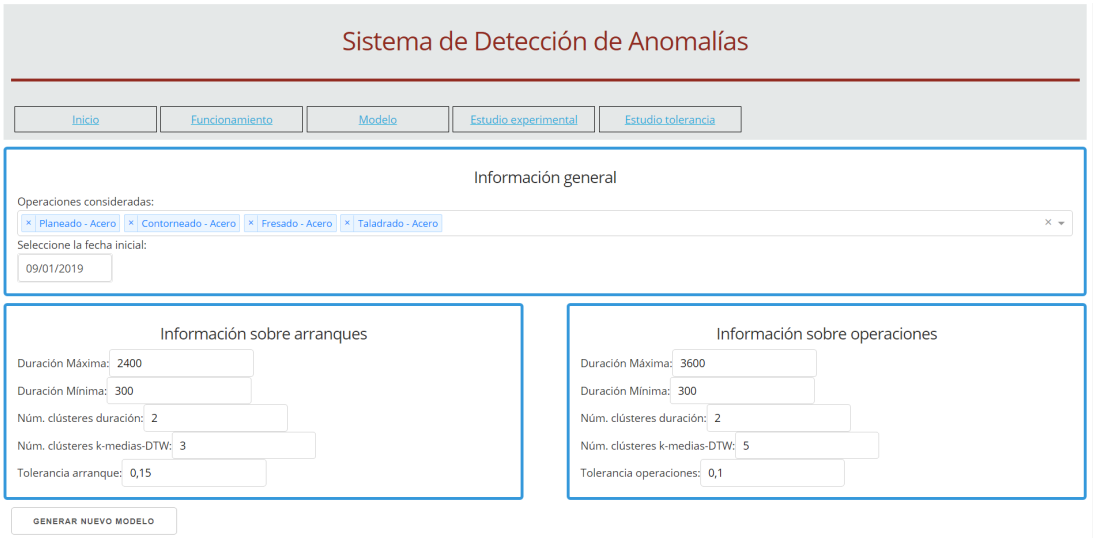


Figura A.5: Página de "Modelo" del SDA

Figura A.6: Página de "Modelo" del SDA con alguna entrada errónea

La opción "Estudio experimental" permite realizar el análisis para determinar la configuración de la clasificación. Se solicita la información de cada campo y se muestra las gráficas del *Capítulo 9*. En esta pestaña, el sistema solicita toda la información necesaria para realizar el estudio: fuente de origen de los datos, información sobre el tipo de operación (identificador de operación, material y duraciones mínimas y máximas) y el rango de configuraciones a estudiar.

La información es introducida en tres etapas (*Figura A.9*), y solo se permite continuar a la siguiente etapa (mediante el correspondiente botón "continuar") cuando los campos introducidos no presentan errores ni de forma ni de dominio de valores. En caso contrario, los errores son notificados, esperando su resolución por parte del usuario (*Figura A.7*).

Cuando la información de todos los campos de entrada supera la validación, y se han completado las tres etapas, el sistema construye y valora los modelos para las configuraciones dadas. Los resultados de las valoraciones con la distancia intra-clúster como función objetivo son representados (*Figura A.10*). Si se detecta algún problema en la conexión o de integridad de la información, entonces el sistema lo notifica (*Figura A.8*).

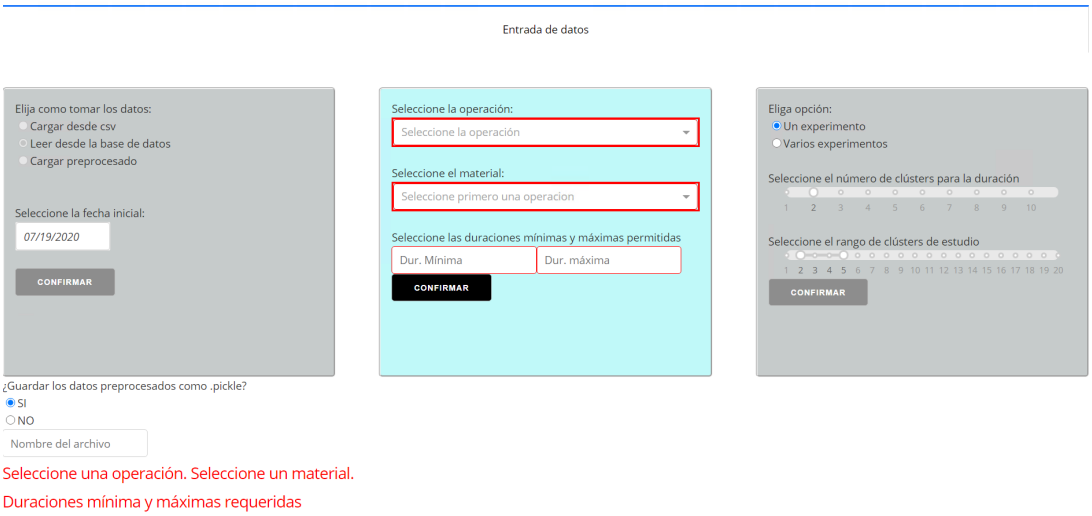


Figura A.7: Página de "Estudio experimental" con errores en los campos de entrada

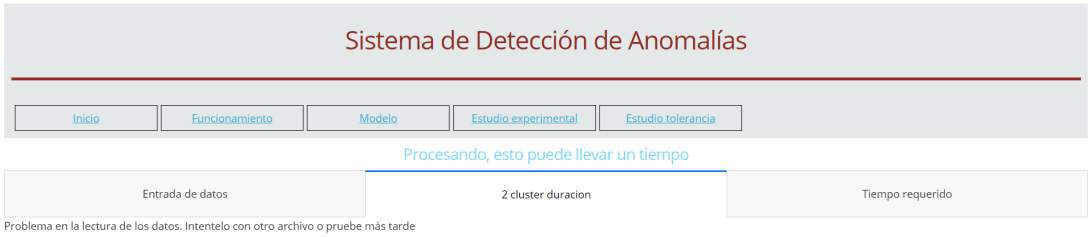


Figura A.8: Página de "Estudio experimental" con problemas en la conexión con la BBDD

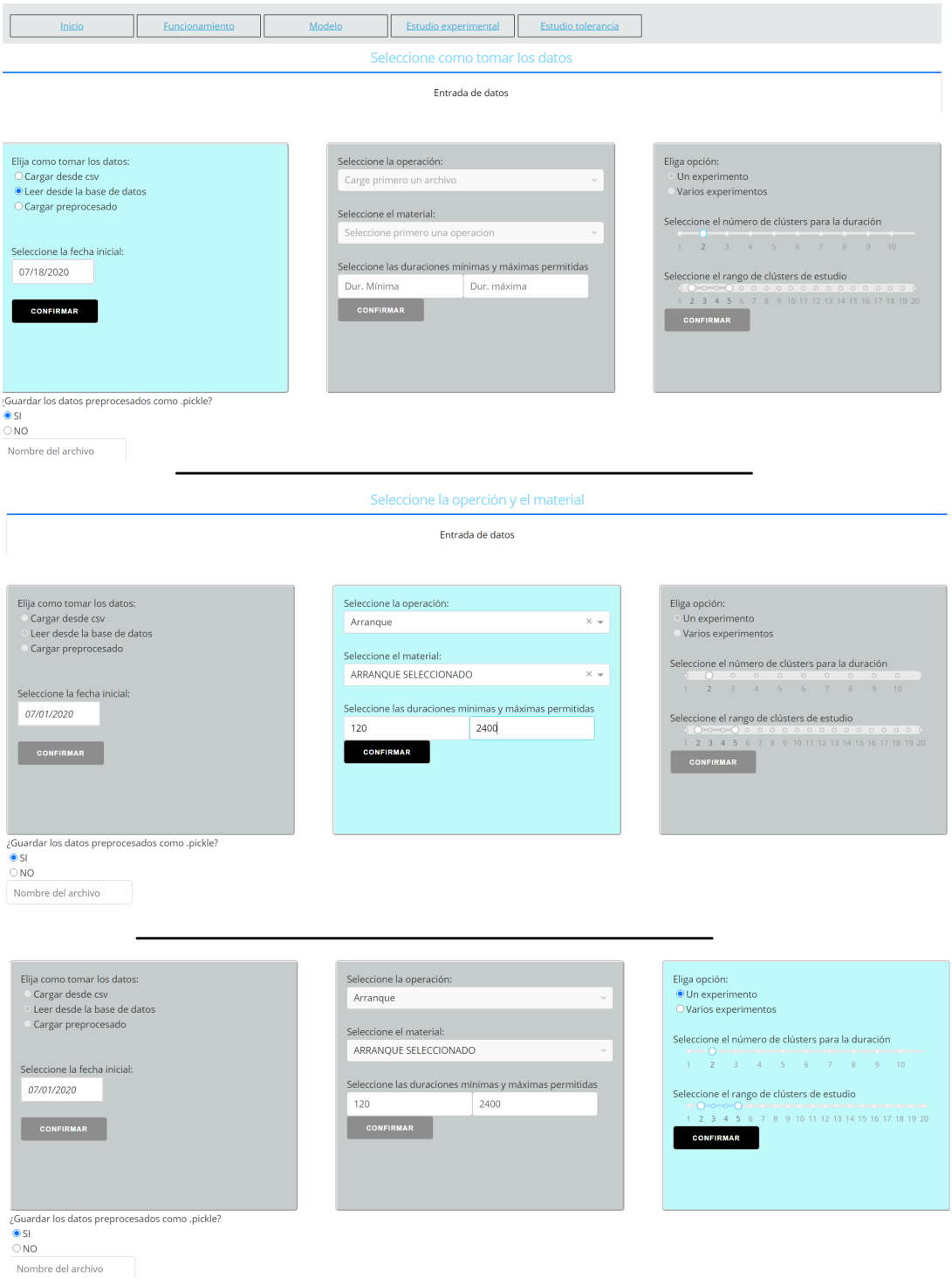


Figura A.9: Página de “Estudio experimental”

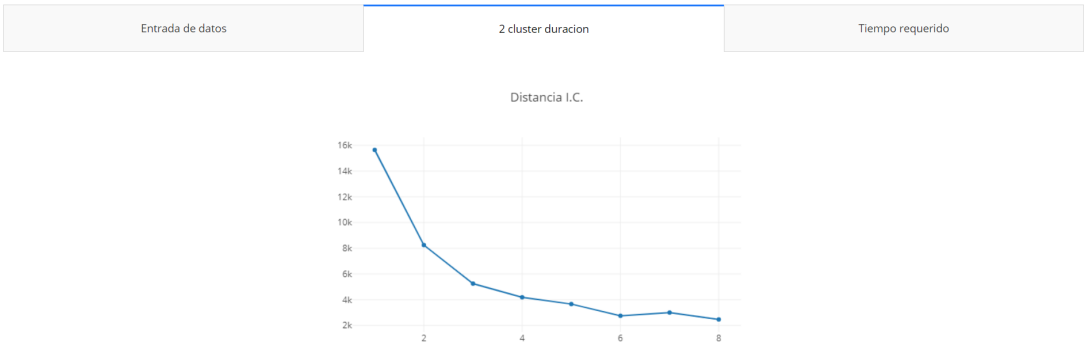


Figura A.10: Interfaz de resultado del estudio experimental en el SDA

Por último, el estudio de la tolerancia es realizado en la opción "estudio tolerancia". En esta página, las gráficas son generadas cuando se modifica alguno de los campos de información. El botón "Cambiar colores" se utiliza para modificar la gama cromática aleatoria con la que se determinan los colores en la representación.

Figura A.11: Página "estudio tolerancia" del SDA

Apéndice B

Implementación de los algoritmos en pseudocódigo

En esta sección se presentan las implementaciones en pseudocódigo correspondientes a los algoritmos presentados en el *capítulo 4* de esta memoria.

B.1. Algoritmo de Medias Móviles

Pseudocódigo del Algoritmo de medias móviles para la versión centrada

```
1  real medias_movil(X: vector[1..n], ventana: integer,
2      pesos: vector[1..n]){
3
4      l := longitud(X)
5      u := (ventana - 1)/2
6      y := vector[1..n]
7
8      //Inicializado el vector de medias móviles
9      for i := 1..n do
10         y[i] = nan  //nan es tipo nulo, no numérico y sin valor
11     end_for
12
13     //Cálculo de las medias móviles
14     for i := (u+1)..(l-u) do
15         suma := 0
16         for j := (i-u)..(i+u) do
17             suma := suma + x[j]*pesos[j]
18         end_for
19         y[i] = suma/ventana
20     end_for
21     return(y)}
```

B.2. Implementación de la DTW versión primera

Pseudocódigo del algoritmo DTW versión primera

```
real DistanciaDTW (x: vector [1..n], y: vector [1..m]) {
DTW: = matriz [1..n, 1..m]

    // Inicialización primer termino
    DTW[1,1] := {distancia(x[1], y[1])

    //Llenado primera fila
    for j := 1..m do
        DTW[1,j] = distancia(x[1], y[j]) + DTW[1,j-1]
    end_for

    //Llenado primera columna
    for i := 1..m do
        DTW[i,1] = distancia(x[i], y[1]) + DTW[i-1,1]
    end_for


    //Llenado del total de la matriz
    for i := 2..n do
        for j := 2..m do
            //Calculo de la distancia entre dos observaciones
            //de la serie
            costo := distancia(x[i], y[j])
            //Calculo del coeficiente de la matriz DTW
            DTW [i, j] := costo +
                + minimo(DTW[i-1, j], DTW[i, j-1], DTW[i-1,j-1])
        end_for
    end_for

    return DTW
}
```

B.3. Implementación del algoritmo para hallar la curva de deformación

Pseudocódigo del algoritmo de cálculo de la curva de deformación

```
1  real curvaDeformacion (A: matriz[1..n, 1..m]){
2      delta := lista_vacia
3      i := n
4      j := m
5
6      // Se supone la existencia de un método asociado a las listas
7      // añadir que recibe un elemento y lo incluye en la lista
8      delta.añadir([i,j])
9
10     while i>1 and j>1 do
11     //Si se está en la primera fila de la matriz
12     if i = 1 do
13         j := j - 1
14
15     //Si se está en la primera columna de la matriz
16     elif j = 1 do
17         i := i - 1
18
19     //Si no se está ni en la primera fila ni columna
20     else:
21         coste_min:=minimo(A[i-1, j-1], A[i-1, j], A[i, j-1])
22         //Elección del par de índices según el valor mínimo
23         if A[i-1, j] = coste_min do
24             i := i - 1
25         elif A[i, j-1] = coste_min do
26             j := j-1
27         else:
28             i := i - 1
29             j := j- 1
30         end_if
31     end_if
32     delta.añadir([i, j])
33     end_while
34     delta.añadir([1,1])
35     return delta.invertir()
36 }
37 //invertir es un método asociado a la lista que invierte el
38 //orden los elementos
```

B.4. Implementación del algoritmo para determinar las asociaciones de índices

Pseudocódigo del algoritmo para determinar las asociaciones

```
1  vector[1..n] asociaciones(phi: lista){
2      //phi es una lista indexada en 1 con P vectores de longitud dos.
3      indices: vector[1..2]
4      phiX: int
5      phiY: int
6      gamma := vector[1..n]
7
8      for i :=1..n
9          gamma[i] = lista_vacia()
10     end_for
11
12     for i := ..P
13         indices := phi[i]
14         phiX := indices[1]
15         phiY := indices[2]
16         gamma[phiX].añadir(phiY)
17     end_for
18     return gamma
19     }
```

B.5. Implementación algoritmo DBA

La implementación en pseudocódigo siguiente se corresponde con el paso tercero del algoritmo DBA, suponiendo determinada la serie **C**, que será el centroide de la iteración anterior en el clústering.

Algoritmo DBA

```
1  vector DBA(C: vector[1..T], S: lista, A: lista,
2  pertenece: vector[1..n], T: int){
3      // C es la serie temporal de la comparación
4      // S es la lista que contiene las series temporales a promediar
5      // A es la lista de asociación. De forma que el elemento A[i][s] es
6      // una lista con el conjunto de índices de la serie i-ésima
7      // que se asocian con el índice s de la serie C.
8      // pertenece es un vector de ceros y unos que indica si el dato
9      // es tenido en cuenta para el cálculo de la media
10     C_prima: vector[1..T]
11
12     for s := 1..T do
13         suma := 0
14         for i:= 1..m do
15             for indice in A[i][s]
16                 suma = suma + S[i] [indice] * pertenece[i]
17             end_for
18         end_for
19         C_prima[j] = suma/m
20     end_for
21     return C_prima
22 }
```

El vector `pertenece` de la implementación anterior se ha incluido para su futura aplicación dentro del algoritmo de k-medias. Pues en dicho caso solo interesa calcular la media DBA de aquellas series temporales que pertenecen al clúster concreto. También cabe destacar que la longitud de la serie muestra viene dada por parámetro por `T` dado que se prefija en la media anterior, `C`

B.6. Implementación del algoritmo de clustering sobre vectores

La implementación presentada se corresponde a la versión dada para según matriz de pertenencia. Se ha separado del porpio algoritmo tres funcionalidades para facilitar su comprensión. Así, en la implementación del algoritmo de k-medias aparece el uso de las funciones `calculo_matriz_pertenencia`, `calculo_centroides` y `calculo_variacion` cuyas implementaciones se muestran a continuación.

Pseudocódigo para el cálculo de la matriz de pertenencia

```
1  matriz[1..m, 1..k] calculo_matriz_pertenencia(
2                                  data: matriz[1..m, 1..n],
3                              centroides: matriz[1..k, 1..n]){
4     //Inicialización matriz de pertenencia
5      for i := 1..m
6          for j := 1..k
7              W[i,j] := 0
8          end_for
9      end_for
10
11     //Cálculo de la matriz de pertenencia
12     for d := 1..m do
13         dist_min := infinito
14         cluster := 1
15         for c := 1..k do
16             for r:= 1..n do
17                 centroide: vector[1..n]
18                 centroide :=  centroides[c,r]
19                 dato := vector[1..n]
20                 dato := data[d,r]
21             end_for
22             // distancia es una función de comparación
23             dist := distancia(centroide, dato)
24             if dist < dist_min
25                 dist_min := dist
26                 cluster := c
27             end_if
28         end_for
29        W[d,cluster] := 1
30     end_for
31     return W
32 }
```

Pseudocódigo para el cálculo de los centroides

```
1  matriz[1..k, 1..n] calculo_centroides(W: matriz[1..m, 1..k],
2                                         data: vector[1..m, 1..n],
3                                         centroides: matriz[1..k, 1..n]){
4      for c := 1..k do
5          for t := 1..n do
6              suma: int
7              suma := 0
8              elementos_cluster: int
9              elementos_cluster := 0
10             for j := 1..m do
11                 suma := W[j,c] * data[j,t]
12                 elementos_cluster := W[j,c]
13             end_for
14             centroides[c,t] := suma/elementos_cluster
15         end_for
16     end_for
17     return centroides
18 }
```

Pseudocódigo para el cálculo de la variación total intra-clúster

```
1  real calculo_variacion(W: matriz[1..m, 1..k],
2                         data: vector[1..m, 1..n],
3                         centroides: matriz[1..k, 1..n]){
4      variacion: int
5      variacion := 0
6      for t :=1..k
7          for i := 1..m
8              for r:= 1..n do
9                  centroide: vector[1..n]
10                 centroide := centroides[c,r]
11                 dato := vector[1..n]
12                 dato := data[d,r]
13             end_for
14             variacion := variacion +
15                          distancia(centroide, dato) * W[i,t]
16         end_for
17     end_for
18     return variacion
19 }
```

Pseudocódigo del algoritmo de k-medias

```
1  matriz[1..m, 1..k] kmeans (data: vector[1..m, 1..n], k: int,
2      tolerancia: int) {
3      //Cada fila de la matriz data es un vector dato de dimensión 1xn
4      W: matriz[1..m, 1..k]
5      W_iteracion_ant: matriz[1..m, 1..k]
6      centroides: matriz[1..k, 1..n]
7      seguir: bool
8      variacion_cluster: int
9
10     seguir := True
11     variacion_cluster := infinito
12     //Inicialización matriz de pertenencia iteracion anterior
13     for i := 1..m
14         for j := 1..k
15             W_iteracion_ant[i,j] := 0
16         end_for
17     end_for
18     // Inicializacion centroides (existe una función para ello)
19     centroides = inicializacion_aleatoria_centroides()
20
21     //Iteraciones
22     while sigue
23     //Asignación de los clústers
24         W := calculo_matriz_pertenencia(data, centroides))
25         //Cálculo de los nuevos centroides
26         centroides =  calculo_centroides(W, data, centroides)
27         //Evaluación de la condición de parada
28         variacion_cluster:= calculo_variacion(W, data,centroides)
29         if W = W_iteracion_ant
30             seguir := False
31         end_if
32         if variacion_cluster < tolerancia:
33             seguir := False
34         end_if
35         W_iteracion_ant := W
36     end_while
37     return W
38 }
```

B.7. Implementación del algoritmo de clustering sobre series temporales

Para la implementación sobre series temporales, se realizan las modificaciones oportunas sobre la implementación en la *sección B.6*, siguiendo el mismo esquema y la misma implementación modular.

Se han de tener en cuenta las observaciones realizadas en la *sección 4.5* sobre la complejidad del cálculo de la distancia DTW y la necesidad de realizar dicho cálculo el mínimo número de veces necesarias. También caben destacar el uso de las funciones `distanciaDTW`, `curvaDeformacion`, `asociaciones` y `DBA` implementadas anteriormente en este *Apéndice*.

Pseudocódigo para el cálculo de los centroides sobre series temporales

```
1  matriz[1..k, 1..n] calculo_centroides(W: matriz[1..m, 1..k],
2      data: lista,
3      centroides: matriz[1..k, 1..T],
4      gammas: lista){
5      m = numero_elementos_en(data)
6      pertenece: vector[1..m]
7      for c := 1..k do
8          for i := 1..m do
9              pertenece := W[i,c]
10         end_for
11         centro := DBA(centroide[c], data, gammas, pertenece, T)
12         for j := 1..T
13             centroides[c,j] := centro[j]
14     end_for
15     return centroides
16 }
```

Pseudocódigo para el cálculo de la variación total intra-clúster

```
1  real calculo_variacion(W: matriz[1..m, 1..k],
2                         distancias: vector[1..m]){
3      variacion: int
4      variacion := 0
5      for t :=1..k
6          for i := 1..m
7              variacion := variacion + distancias[i] * W[i,t]
8          end_for
9      end_for
10 return variacion
11 }
```

Pseudocódigo para el cálculo de la matriz de pertenencia sobre series temporales

```
1  (matriz[1..m, 1..k], vector[1..T]) calculo_matriz_pertenencia(
2       data: lista, centroides: matriz[1..k, 1..T]){
3      //Inicialización matriz de pertenencia
4      m := numero_elementos_en(data)
5      for i := 1..m
6          for j := 1..k
7              W[i,j] := 0
8          end_for
9      end_for
10     gammas: lista_vacia
11     distancias: vector[1..m]
12     //Cálculo de la matriz de pertenencia
13     for d := 1..m do
14         dist_min := infinito
15         cluster: int
16         dato: vector
17         dato: data[d]
18         L: int
19         L := longitud(dato)
20         for c := 1..k do
21             for r:= 1..n do
22                 centroide: vector[1..T]
23                 centroide :=  centroides[c,r]
24             end_for
25             ADTW := distanciaDTW(centroide, dato)
26             dist = ADTW[L,T]
27             if dist < dist_min
28                 dist_min := dist
29                 cluster := c
30                 A_DTW_cluster = ADTW
31             end_if
32         end_for
33         distancias[d] = dist_min
34         W[d,cluster] := 1
35         phi := curvaDeformacion(A_DTW_cluster)
36         gammas.añadir(identificacionIndices(phi))
37     end_for
38     return (W, gammas, distancias)
39 }
```

Pseudocódigo del algoritmo de k-medias

```
matriz[1..m, 1..k] kmeans (data: lista, k: int,
                           tolerancia: int) {
//Cada fila de la matriz data es un vector dato de dimensión 1xn
    m: int
    m := numero_elementos_en(data)
    W: matriz[1..m, 1..k]
    W_iteracion_ant: matriz[1..m, 1..k]
    centroides: matriz[1..k, 1..n]
    seguir: bool
    variacion_cluster: int

    seguir := True
    variacion_cluster := infinito
    //Inicialización matriz de pertenencia iteracion anterior
    for i := 1..m
        for j := 1..k
        W_iteracion_ant[i,j] := 0
        end_for
    end_for
    // Inicializacion centroides (existe una función para ello)
    centroides = inicializacion_aleatoria_centroides()

    //Iteraciones
    while seguir
        //Asignación de los clústers
        (W,gammas, distancias) := calculo_matriz_pertenencia(
                                                data, centroides)
        //Cálculo de los nuevos centroides
        centroides =  calculo_centroides(W, data,
                                         centroides, gammas)
        //Evaluación de la condición de parada
        variacion_cluster:= calculo_variacion(W, distrancias)
        if W == W_iteracion_ant
            seguir := False
        end_if
        if variacion_cluster < tolerancia:
            seguir := False
        end_if
        W_iteracion_ant := W
    end_while
    return W
}
```

Apéndice C

Información complementaria al fundamento teórico sobre series temporales

C.1. El concepto de serie temporal

En esta sección se introducen los conceptos fundamentales sobre series temporales (*Definiciones C.1, C.2 y C.9*), así como las principales técnicas y modelos que se aplicaran en el análisis de series temporales. En particular, se centrará en el estudio de un modelo de descomposición de series temporales denominado *modelo aditivo* (*Def. C.7*) y en las técnicas para determinar la *tendencia* (*Def. C.4*) de una serie temporal. Esta primera sección se basa en los capítulos primero y segundo de [2] y en los capítulos octavo y noveno de [3]. Este apéndice se complementa con el *Capítulo segundo* de [25], donde se pueden encontrar las demostraciones a las afirmaciones realizadas.

C.1.1. Introducción

En esta sección se definen los conceptos de serie temporal y su modelo (*Def. C.2*) y se concluye con una primera clasificación de las series temporales.

Definición C.1. *Se define un* ***proceso estocástico*** *o proceso aleatorio como una familia de variables aleatorias* $\{X_i : \ i \in T, \ X_i : \Omega \to \mathcal{S}\}$*, con* $T \subseteq \mathbb{R}$*, que representan el espacio muestral* Ω *en el conjunto* $\mathcal{S}$*.*
Al conjunto T *se le denomina* ***conjunto de índices*** *y a* $\mathcal{S}$ *se le denomina* ***espacio de estados****.*

Existen diferentes formas de tomar tanto el conjunto de índices como el espacio de estados. Generalmente, $\mathcal{S}$ suele ser el conjunto de los reales $\mathbb{R}$ y T se toma como un conjunto discreto (los naturales $\mathbb{N}$) o como un intervalo.

Definición C.2. *Sea* $\{X_i : \ i \in T, \ X_i : \Omega \to \mathcal{S}\}$ *un proceso estocástico, una* ***serie temporal*** *es una realización de cada una de las variables aleatorias que componen el proceso estocástico. Es decir, es una familia de la forma*

$$\boldsymbol{X} = \{X_i(\omega_i) : \ i \in T, \ \omega_i \in \Omega\}.$$

En esta situación, se dice que el proceso estocástico $\{X_i : \ i \in T, \ X_i : \Omega \to \mathcal{S}\}$ *es el* ***modelo de la serie temporal*** $\boldsymbol{X}$*.*

Por tanto, una serie temporal es un conjunto de observaciones x_t recogidas en un tiempo específico t según un modelo que es un proceso estocástico. Entonces el conjunto de índices T índica los momentos en el que se realizan las mediciones. Dependiendo de cómo se tome dicho conjunto se tienen dos tipos de series temporales.

Definición C.3. *Sea* $\boldsymbol{X} = \{X_i(\omega_i) : \ i \in T, \ \omega_i \in \Omega\}$ *una serie temporal,*

1. *Se dice que* $\boldsymbol{X}$ *es una serie temporal (o un proceso estocástico) es de* ***tiempo discreto*** *si el conjunto de índices* T *es discreto.*
2. *Se dice que* $\boldsymbol{X}$ *es una serie temporal (o un proceso estocástico) es de* ***tiempo continuo*** *si el conjunto de índices* T *es un intervalo de la recta real.*

C.1.2. Componentes de una serie temporal: El modelo aditivo

El modelo básico y más general, divide una serie temporal en tres componentes (que a su vez son una serie temporal cada una de ellas) y que son la tendencia (*trend*), la estacionalidad (*seasonality*) y el residuo (*residual*). En esta sección, se suponen series temporales de tiempo discreto cuyo conjunto de índices está contenido en $\mathbb{N}$ y tiene cardinal finito.

Definición C.4. *Sea* $\boldsymbol{X} = [x_1, \ldots, x_n]$ *una serie temporal:*

1. *Se define la* ***tendencia*** *(trend) como el comportamiento o movimiento regular (monotonía) de la serie.*
2. *Se define la* ***estacionalidad*** *(seasonality) como los movimientos de oscilación de la serie.*
3. *Se define el* ***residuo****, residual o resto (residual) como las variaciones aleatorias restantes que no se pueden englobar en la tendencia o en la estacionalidad.*

Notación. Como notación general, $\mathbf{T} = [t_1, \ldots, t_n]$, $\mathbf{S} = [s_1, \ldots, s_n]$ y $\mathbf{R} = [r_1, \ldots, r_n]$ denotan respectivamente la tendencia, estacionalidad y el residuo de la serie temporal $\mathbf{X} = [X_1, \ldots, X_n]$.

Entonces, dada una serie temporal la tendencia comprende la evolución global de la serie y la estacionalidad captura los efectos de los eventos que ocurren con cierta repetición en el tiempo, tales como los eventos periódicos. Por último, el residuo recoge todo aquello que no puede ser capturado ni por la tendencia ni por la estacionalidad.

Ejemplo C.5. El siguiente ejemplo ilustra el concepto de tendencia. Para ello se supone una serie temporal definida de la siguiente forma:

$$x_t = t + 2 \cdot sen(t) \quad t = 0 \dots 50$$

Dado que $-2 \leq 2sen(t) \leq 2 \quad \forall t$, entonces se tiene que $x_t \leq x_{t+3}$ luego la tendencia en su conjunto es creciente. La *Figura C.1* ilustra el ejemplo.

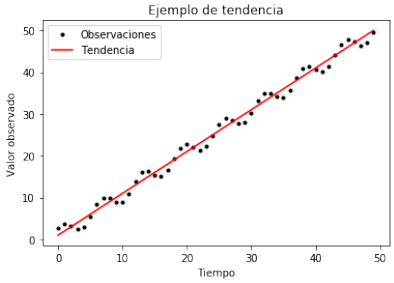


Figura C.1: Ejemplo de la tendencia de una serie temporal

Dependiendo de la duración de las series temporales, generalmente cuando estas comprenden varios años, la estacionalidad se divide en variaciones estacionales y variaciones cíclicas. Las **variaciones estacionales** son oscilaciones a corto plazo mientras que las **variaciones cíclicas** son los movimientos a largo plazo.

Ejemplo C.6. Como ejemplo, supongamos que se realiza mediciones de la temperatura en una estación meteorológica concreta cada hora del día durante diez años. Si se analiza cada año se observa una periodicidad de la temperatura media diaria que aumenta en los meses de primavera, alcanza su máximo en verano y disminuye en los meses de otoño hasta alcanzar su mínimo en invierno. Este comportamiento que se verifica todos los años se correspondería con las variaciones cíclicas. Finalmente, si se observa un día concreto, se tiene que la temperatura en cada hora aumenta desde el amanecer a la puesta de sol y disminuye desde el anochecer hasta que amanece. Estas oscilaciones que se repiten de forma diaria se incluirían por las variaciones estacionarias.

En el caso de estudio, este tipo de variaciones a largo plazo carece de sentido y por tanto se omitirán en lo que resta, considerando la estacionalidad como las variaciones estacionales.

El modelo básico de descomposición sobre series temporales es el **modelo aditivo**. Este modelo supone que la serie temporal es la suma de sus componentes.

Definición C.7. *sea* $\boldsymbol{X} = [x_1, \ldots, x_n]$ *una serie temporal y sean* $\boldsymbol{T} = [t_1, \ldots, t_n]$, $\boldsymbol{S} = [s_1, \ldots, s_n]$ *y* $\boldsymbol{R} = [r_1, \ldots, r_n]$ *la tendencia, la estacionalidad y el residuo respectivamente, se define el* ***modelo aditivo*** *como aquel que verifica las siguientes hipótesis:*

1. *La serie original es la suma de sus componentes. Esto es:*

$$x_k = t_k + s_k + r_k \quad \forall k = 1, \ldots, n \tag{C.1}$$

2. *Los modelos de las componentes de la serie temporal* $\boldsymbol{T}, \boldsymbol{S}, \boldsymbol{R}$ *son independientes entre sí para cada índice fijo, y el modelo de la serie* $\boldsymbol{R}$ *es débilmente estacionario.*

Ejemplo C.8. Como ejemplo se presenta una serie temporal que cumple el modelo aditivo. Con las notaciones anteriores:

$$x_k = 2,75k + 10sen(k) + 10 \quad k = 1, 2, \ldots, 50$$

La serie temporal resultante se representa en la *Figura C.2*. Claramente, por las definiciones dada en la *Definición C.4* se tiene que la tendencia viene dada por $t_k = 2{,}75k + 10$ y la estacionalidad viene dada por $s_k = 10sen(k)$. Con la construcción definida se cumple que no hay residual, puesto que no hay valores aleatorios que no sean parte ni de la tendencia ni de la estacionalidad, luego $r_k = 0$. En la *Figura C.2* se puede observar la descomposición.

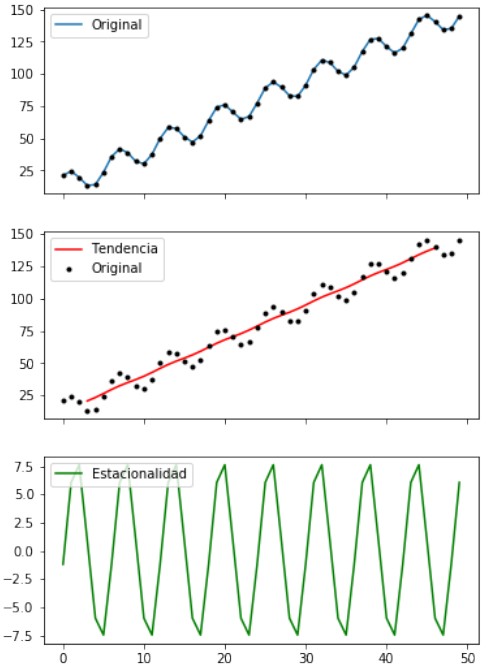


Figura C.2: Serie temporal definida como $x_k = 2,75k + 10sen(k) + 10 \quad k = 1, 2, \ldots, 51$

Existe un segundo modelo que supone que la serie es el producto de sus componentes. Este modelo se denomina **modelo multiplicativo**, y con las notaciones anteriores se expresa como:

$$x_k = t_k \cdot s_k \cdot r_k \quad \forall k = 1, \ldots, n \tag{C.2}$$

Se dice que este modelo deriva del modelo aditivo puesto que es equivalente a aplicar las hipótesis de dicho modelo a la transformación por el logaritmo de la serie original. Sea $\mathbf{X} = [x_1, \ldots, x_n]$ una serie temporal que verifica una descomposición multiplicativa (C.2) entonces:

$$log(x_k) = log(t_k) + log(s_k) + log(r_k) \quad \forall k = 1, \ldots, n$$

C.1.3. Series temporales con modelo estacionario

En esta sección se va a estudiar un tipo de serie temporal cuyo modelo presenta unas características muy concretas que facilitan su estudio.

Notación. Sean $X, Y : \Omega \to \mathbb{R}$ dos variables aleatorias reales, se denotan la esperanza de X por $\mathbb{E}(X)$, la varianza de X como $Var(X) = \mathbb{E}((X - \mathbb{E}X)^2)$ a la varianza de X y la covarianza de X e Y por $Cov(X, Y) = \mathbb{E}((X - \mathbb{E}X)(Y - \mathbb{E}Y))$.

Definición C.9. *Sea $\{X_i : \ i \in T\}$ un modelo de serie temporal con $\mathbb{E}(X_i^2)$ finita para cada $i \in T$, se dice que el modelo es **débilmente estacionario** (o simplemente **estacionario**) si sí existen $\mu, \sigma \in \mathbb{R}$ y un subconjunto real $\{a_t : t \in T\}$ tales que verifican:*

- *Media constante: $\mathbb{E}(X_t) = \mu \ \forall t \in T$*
- *Varianza constante: $Var(X_t) = \sigma \ \forall t \in T$*
- *Autocovarianza: $Cov(X_t, X_{t+h}) = a_h \ \forall t \in T, \forall h \neq 0$ tal que $t + h \in T$*

*En caso de que alguna de las anteriores condiciones no se verifique, se dice que la serie es **no estacionaria**.*

Con abuso del lenguaje, se dice que una serie es estacionaria cuando así lo sea su modelo.

Ejemplo C.10. *Ruido blanco (white noise).* El ejemplo básico de serie estacionaria es el *ruido blanco*, que se caracteriza por tener modelo $\{X_i : \ i = 1 \ldots n\}$ con medias y covarianzas constantemente nulas y varianza constante. El caso más usual es cuando $X_1, ...X_n$ son variables aleatorias independientes e igualmente distribuidas $N(0, \sigma^2)$.

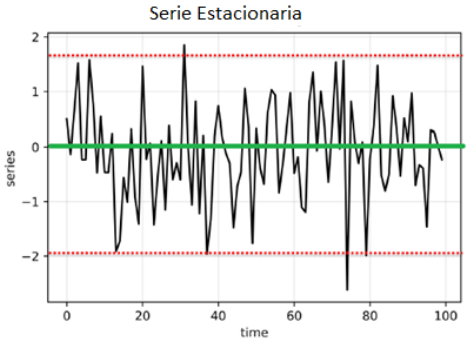


Figura C.3: Ejemplo de serie temporal con modelo estacionario de ruido blanco

Ejemplo C.11. *Camino aleatorio (random walk).* El camino aleatorio $\{S_t : t = 0, 1, ...\}$ se obtiene de la acumulación de variables aleatorias independientes e igualmente distribuidas (vaiid). Es decir, sea $(X_n)_{n=1}^{\infty}$ una sucesión de variables aleatorias independientes e igualmente distribuidas entonces

$$S_n = \sum_{i=1}^{n} X_i$$

Supongamos que X_i está distribuida como una Bernoulli de parámetro $\frac{1}{2}$, $(B(\frac{1}{2}))$, esto es,

$$P(X_i = 0) = \frac{1}{2} \quad \text{y} \quad P(X_i = 1) = \frac{1}{2}.$$

Entonces S_n es una binomial de parámetros n y $p = \frac{1}{2}$, $(b(n, \frac{1}{2}))$ y $\mathbb{E}(S_n) = np = \frac{n}{2}$. Luego la esperanza no es constante y el modelo de la serie no es estacionario.

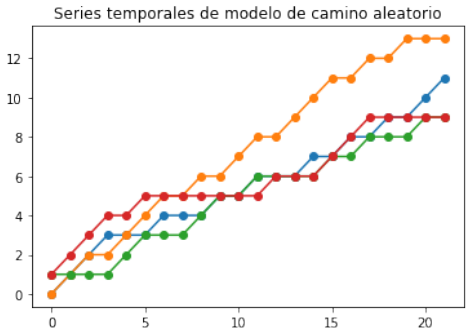


Figura C.4: Ejemplo de series temporales con modelo no estacionario de camino aleatorio

Todo modelo de una serie temporal posee dos funciones características propias. Sea $\{X_i : i \in T\}$ con $T \subseteq \mathbb{Z}$ un modelo de serie temporal con $\mathbb{E}(X_i^2)$ finita para cada $i \in T$, se definen:

1. la **función de medias** $\mu : T \longrightarrow \mathbb{R}$ como

$$\mu(t) = \mathbb{E}(X_t) \quad \forall t \in T$$

2. la **función de covarianzas** como $\gamma : T \times T \longrightarrow \mathbb{R}$ como

$$\gamma(t,s) = Cov(X_t, X_s) \quad \forall t,s \in T$$

Al índice $t_0 \in T$ elegido se le denomina **momento inicial**.

En particular, es condición necesaria y suficiente para que un modelo $\{X_i : \ i \in T\}$ con función de medias μ y función de covarianzas γ sea estacionario que se verifique que:

1. $\mu(t)$ es constante.
2. $\gamma(t, t+h)$ no depende del valor de t para cada $h > 0$ tal que $t + h \in T$.

En consecuencia, todo modelo estacionario $\{X_i : \ i \in T\}$ se le puede caracterizar por las siguientes aplicaciones, que resultan de una simplificación de las aplicaciones características anteriores. Suponemos que $t_0 \in T$, y se definen

1. la **función de autocovarianzas** $\gamma : \{h \in \mathbb{R} \text{ con } t_0 + h \in T\} \longrightarrow \mathbb{R}$ como

$$\gamma(h) = Cov(X_{t_0}, X_{t_0+h})$$

2. la **función de autocorrelación** $\rho : \{h \in \mathbb{R} \text{ con } t_0 + h \in T\} \longrightarrow \mathbb{R}$ como

$$\rho(h) = \frac{\gamma(h)}{\gamma(t_0)}$$

C.1.4. Técnicas para determinar la tendencia. El algoritmo de medias móviles

Bajo la hipótesis del modelo aditivo, la tendencia es una componente clave que permite analizar el comportamiento de la serie temporal obviando los efectos estacionarios y las interferencias anómalas. Para la estimación se diferencian dos grupos de series temporales: series con tendencia determinista y series con tendencia evolutiva. Esta sección está basada en la Sección 9.2 de [3] y en [4].

Definición C.12.

1. *Se dice que una serie temporal tiene* ***tendencia determinista*** *si la tendencia está determinada por una función simple continua y monótona en todo el conjunto de índices.*
2. *En caso contrario, se dice que la serie temporal tiene* ***tendencia evolutiva****.*

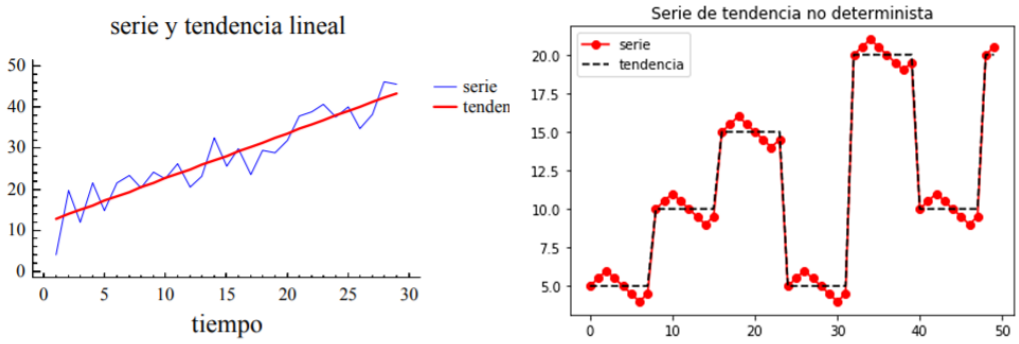


Figura C.5: Ejemplo de serie temporal con tendencia determinista (izquierda) y con tendencia evolutiva (derecha)

Una tendencia evolutiva se caracteriza por variar su monotonía en el tiempo, y se puede aproximar localmente (en un intervalo pequeño de tiempo) por funciones simples y monótonas.

Estimación de tendencia determinista.

En *las series temporales con tendencia determinista* la principal técnica que se aplica en estos casos es el **análisis de regresión**. Este método supone que los parámetros que definen la tendencia permanecen inmutables al tiempo, y busca ajustar la "nube de puntos"por una función continua y monótona. Si $\mathbf{T} = [t_1, \ldots, t_n]$ denota la tendencia, las técnicas de regresión más comunes son:

- La **regresión lineal** supone que la dependencia de tendencia respecto al tiempo viene determinada por un polinomio lineal.
$$t_k = a + bk \qquad a, b \in \mathbb{R}$$
- La **regresión polinómica** supone que la dependencia de la tendencia respecto al tiempo está determinada por un polinomio de grado m. En especial, destaca el uso de la regresión cuadrática ($m = 2$) y cúbica ($m = 3$).
$$t_k = a_0 + \sum_{i=1}^{m} a_i k^i \qquad a_0 \ldots a_m \in \mathbb{R}$$
- La **regresión logarítmica** supone que la tendencia tiene un comportamiento potencial o logarítmico.
$$t_k = a + b \cdot ln(k) \qquad a, b \in \mathbb{R}$$
- La **regresión exponencial** supone que la tendencia tiene un crecimiento exponencial respecto al tiempo.
$$t_k = a + b \cdot e^k \qquad a, b \in \mathbb{R}$$

C.2. La distancia DTW

C.2.1. Caminos de deformación y su coste

En esta sección se va a presentar una medida de comparación que permita trabajar con series temporales de distinta longitud y sea capaz de detectar la deformación en el tiempo. Para ello se va a estudiar las posibles asociaciones entre los índices de dos series temporales buscando la asociación óptima en sentido que minimice la distancia de un elemento de la serie temporal muestra respecto a serie temporal prueba.

Definición C.13. *sean* $\boldsymbol{X} = [x_1, \ldots, x_n]$ *e* $\boldsymbol{Y} = [y_1, \ldots, y_m]$ *dos series temporales con conjuntos de índices* $\{1, \ldots, n\}$ *y* $\{1, \ldots, m\}$ *se define un* ***camino de deformación*** *(warping path) de orden* $n \times m$ *y cardinal* P *como una secuencia de* P *puntos bidimensionales*

$$\{(i_1, j_1), \ldots, (i_P, j_P)\} \subset \{1 \ldots n\} \times \{1 \ldots m\}$$

tales que satisfacen

1. *Condiciones en los bordes:* $(i_1, j_1) = (1, 1)$ *y* $(i_P, j_P) = (n, m)$
2. *Condiciones de monotonía:*
$$i_1 \leq i_2 \leq \ldots \leq i_P$$
$$j_1 \leq j_2 \leq \ldots \leq j_P$$
3. *Condición sobre el tamaño de paso:*
$$(i_s, j_s) - (i_{s-1}, j_{s-1}) \in \{(0,1), (1,0), (1,1)\} \ \forall s = 2 \ldots P$$

Las condiciones de monotonía y de tamaño de paso suponen que, con las notaciones de la definición, para cada $s = 2 \ldots P$ se verifica

$$i_s = \begin{cases} i_{s-1} \\ i_{s-1} + 1 \end{cases}$$

$$j_s = \begin{cases} j_{s-1} \\ j_{s-1} + 1 \end{cases}$$

Un camino de deformación

$$\{(i_1, j_1), \ldots, (i_P, j_P)\} \subset \{1 \ldots n\} \times \{1 \ldots m\}$$

se denota como una aplicación $\Psi : \{1 \ldots P\} \longrightarrow \{1 \ldots n\} \times \{1 \ldots m\}$ tales que para cada $s \in \{1 \ldots P\}$ se verifica que $(i_s, j_s) = \Psi(s)$. Además, con un abuso de notación, se denota al conjunto igual que a la aplicación definida, de forma que la situación $(i, j) = \Psi(s)$ para algún s

se representa como $(i, j) \in \Psi$. Además, a cada componente de Ψ se les denota respectivamente como ψ_X y ψ_Y, de forma que $\Psi = (\psi_X, \psi_Y)$.

Geométricamente, un camino de deformación establece una asociación entre índices de dos series temporales. La asociación básica, que es la que utiliza la distancia euclídea, es el momento de ocurrencia, pero con un camino de deformación se pueden establecer asociaciones de índices con momento de ocurrencia distinto. Esto es lo que le da el adjetivo "deformación".

Así pues, sea $\Psi = \psi_x, \psi_Y)$ un camino de deformación de orden $n \times m$ y cardinal P, se puede considerar una nueva línea temporal (conjunto de índices) $\{1 \dots P\}$ y dos series temporales de igual longitud $\mathbf{X}' = [x'_1, \dots, x'_P]$ e $\mathbf{Y}' = [y'_1, \dots, y'_P]$ tales que

$$\text{para cada } s = 1 \dots P \quad x'_s = x_{\psi_X(s)} \text{ e } y'_s = y_{\psi_Y(s)}$$

Y considerar la asociación dada por el momento de ocurrencia. Nótese que $\mathbf{X}'$ e $\mathbf{Y}'$ son el resultado de "deformar" las series originales según Ψ, y se dice que son extensiones de $\mathbf{X}$ e $\mathbf{Y}$. En la *Definición C.18* se detalla el concepto de extensión y en la *Sección C.2.3* se profundiza en esta idea.

Las asociaciones de índices dadas por un camino de deformación satisfacen por definición una restricción temporal lógica, que se enuncia como "*el tiempo no puede retroceder solo detenerse*" y es lo que indica la condición segunda de la *Definición C.13*. Además, las condiciones segunda y tercera garantizan que Ψ, como aplicación, es inyectivo, lo que se enuncia como "*el tiempo no puede detenerse en ambas series*". Finalmente, las condiciones en los bordes garantizan que:

1. Para cada $i \in \{1 \dots n\}$ existen $j \in \{1 \dots m\}$ y $s \in \{1 \dots P\}$ tales que $\Psi(s) = (i, j)$.
2. Para cada $j \in \{1 \dots m\}$ existen $i \in \{1 \dots n\}$ y $s \in \{1 \dots P\}$ tales que $\Psi(s) = (i, j)$.

Esto se enuncia como "*el tiempo no puede saltar*".

Notación. Se denota al conjunto de los caminos de deformación orden $n \times m$ de cardinal no fijo se los denota por $\mathcal{P}_{n,m}$.

Si P es el cardinal de un camino de $\mathcal{P}_{n,m}$ entonces se verifica que

$$\text{máx}\{n, m\} \leq P \leq n + m$$

Definición C.14. *sean* $\boldsymbol{X} = [x_1, \dots, x_n]$ *e* $\boldsymbol{Y} = [y_1, \dots, y_m]$ *dos series temporales con conjuntos de índices* $\{1, \dots, n\}$ *y* $\{1, \dots, m\}$ *respectivamente, sea* $\mathbb{V}$ *un espacio métrico que contiene a todos los coeficientes de las series temporales dotado de una distancia* d*, y sea* Ψ *un camino por deformación de orden* $n \times m$ *y cardinal* P*, se define el coste del camino para las series* $\boldsymbol{X}$ *e* $\boldsymbol{Y}$ *como*

$$C_{\boldsymbol{X},\boldsymbol{Y}}(\Psi) = \sum_{(i,j)\in\Psi} d(x_i, y_j) = \sum_{i=1}^{P} d(x_{\Psi_X(i)}, y_{\Psi_Y(j)})$$

C.2.2. La distancia DTW como coste mínimo

En esta sección se define la distancia DTW (*Dynamic Time Warping*) de dos series temporales y se estudian sus principales propiedades, así como el camino de deformación óptimo en el sentido de coste mínimo (véanse [9] y [12]).

Se supone que $\mathbb{V}$ es un espacio métrico (generalmente $\mathbb{R}$) con una distancia denotada por d y que contiene a todos los coeficientes de las series temporales que se presentan. Estas series son de tiempo discreto con su conjunto de índices contenido en los naturales y de cardinal finito.

Definición C.15. *sean* $\boldsymbol{X} = [x_1, \ldots, x_n]$ *e* $\boldsymbol{Y} = [y_1, \ldots, y_m]$ *dos series temporales con* $\boldsymbol{X}$ *la serie muestra e* $\boldsymbol{Y}$ *la serie prueba:*

1. *Se define la* ***distancia DTW*** *(Dynamic Time Warping) de las series temporales* $\boldsymbol{X}$ *frente a* $\boldsymbol{Y}$ *como*

$$d_{DTW}(\boldsymbol{X}, \boldsymbol{Y}) = \min_{\Psi \in \mathcal{P}_{n,m}} \left(\sum_{(i,j)\in\Psi} d(x_i, y_j) \right) = \min_{\Psi \in \mathcal{P}_{n,m}} C_{\boldsymbol{X},\boldsymbol{Y}}(\Psi) \tag{C.3}$$

2. *Se define la* ***curva de deformación*** Φ *como el camino de deformación que hace mínimo la expresión (C.3). Es decir,*

$$\Phi = \arg\min_{\Psi \in \mathcal{P}_{n,m}} \left(\sum_{(i,j)\in\Psi} d(x_i, y_j) \right) = \arg\min_{\Psi \in \mathcal{P}_{n,m}} C_{\boldsymbol{X},\boldsymbol{Y}}(\Psi) \tag{C.4}$$

Es decir, la distancia DTW es el coste del camino de deformación de coste mínimo asociado a dichas series. El concepto de curva de deformación permite formalizar la definición de deformación en el tiempo.

Definición C.16. *Sean* $\boldsymbol{X} = [x_1, \ldots, x_n]$ *e* $\boldsymbol{Y} = [y_1, \ldots, y_m]$ *dos series temporales y sea* $\Phi = (\phi_X, \phi_Y)$ *la curva de deformación, se dice que se ha producido una* ***deformación en el tiempo*** *en la comparación de* $\boldsymbol{X}$ *frente a* $\boldsymbol{Y}$ *en el momento* $k+1$ *si se da alguno de los siguientes casos:*

(a) $\phi_X(k+1) = \phi_X(k) \quad y \quad \phi_Y(k+1) = \phi_Y(k) + 1$

(b) $\phi_X(k+1) = \phi_X(k) + 1 \quad y \quad \phi_Y(k+1) = \phi_Y(k)$

En el caso (a) se dice que se ha producido una ***inserción*** *y en el caso (b) se dice que se ha producido una* ***eliminación****.*
Además, si se verifica que

(c) $\phi_X(k+1) = \phi_X(k) + 1 \quad y \quad \phi_Y(k+1) = \phi_Y(k) + 1$

entonces se dice que se ha producido una ***coincidencia temporal*** *en* $k+1$*.*

Ejemplo C.17. Consideramos las series temporales $\mathbf{X} = [3,3,6,0,3,3]$ e $\mathbf{Y} = [3,3,3,6,0,3]$. Su distancia euclídea es de $\sqrt{54}$, y su distancia como suma de las distancias siguiendo el orden cronológico es de 12 (*Fig. C.6 A*). Sin embargo, si consideramos la asociación de puntos presente en la *Figura C.6 B*) dada por el camino de deformación

$$\Psi = \{(1,1),(2,2),(2,3),(3,4),(4,5),(5,6),(6,6)\}$$

El coste de dicho camino para la comparación de las series temporales es cero.

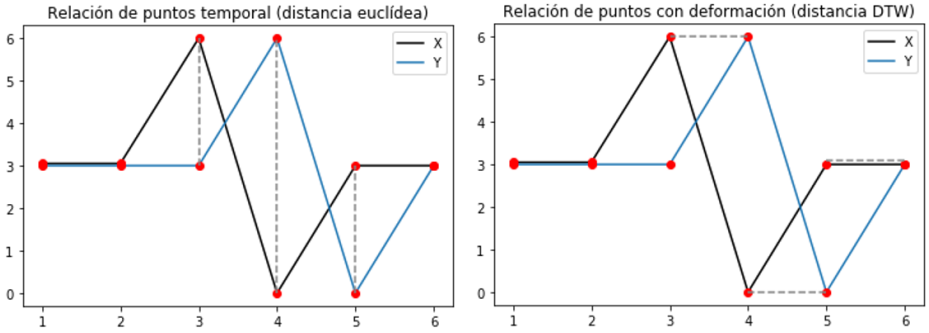


Figura C.6: Comparación de la asociación cronológica (A) con una asociación con deformación en el tiempo (B). Las líneas grises punteadas indican las asociaciones.

C.2.3. Comportamiento de la distancia DTW frente a deformaciones

En esta sección se presenta el resultado que justifica que si la serie temporal $\mathbf{Y}$ es una extensión de la serie $\mathbf{X}$ entonces el valor $d_{DTW}(\mathbf{X},\mathbf{Y})$ es cero, y en consecuencia la distancia DTW las identifica como "iguales". Es más, la propiedad fundamental de la distancia *DTW* se extiende a la comparación de dos series temporales que sean deformaciones de una misma serie original. Todos estos conceptos se presentan en esta sección. Los detalles de las afirmaciones de esta sección se pueden consultar en [25]

La relación fundamental entre dos series temporales es el efecto de deformación en el tiempo. Analíticamente este efecto se observa cuando aparecen repetidos de forma consecutiva algún término de la serie temporal (redundante), como recoge la siguiente definición.

Definición C.18. *Se dice que una serie* $\boldsymbol{Y} = [y_1,\ldots,y_m]$ *es una* ***extensión por deformación*** *(o simplemente extensión) de la serie* $\boldsymbol{X} = [x_1,\ldots,x_n]$ *si:*

1. *Para cada* $i \in \{1\ldots n\}$ *existe* $j \in \{i\ldots m\}$ *tal que* $x_i = y_j$.
2. *Se puede establecer una identificación entre los índices* $\gamma : \{1\ldots m\} \longrightarrow \{1\ldots n\}$ *tal que:*
   a) $\gamma(t) \leq \gamma(t+1)\ \ \forall t = 1..m$
   b) $\gamma(1) = 1$

c) $\gamma(m) = n$

d) $y_t = x_{\gamma(t)}$

Se denota por $\boldsymbol{Y} \gg \boldsymbol{X}$ *el caso en que* $\boldsymbol{Y}$ *es extensión de* $\boldsymbol{X}$.
Además, se dice que una serie temporal $\boldsymbol{Z}$ *está en* ***forma condensada*** *si no es extensión de ninguna otra serie temporal distinta a sí misma.*

Es claro que las condiciones de extensión implican que $m \geq n$. Con esta idea, se tiene que una serie temporal $\mathbf{X} = [x_1, \ldots, x_n]$ es forma condensada si y solo si $x_i \neq x_{i+1},\ i = 1...n-1$. En consecuencia, para obtener la forma condensada de una serie temporal arbitraria basta con "ignorar" los coeficientes que aparecen repetidos de forma consecutiva. Con ello se justifica la existencia y unicidad de la forma condensada.

A la acción de "*suprimir términos repetidos*" se le denomina **condensación** y a la acción de "*duplicar términos consecutivos*" se le denomina **extensión**. Estas son la dos únicas posibles deformaciones temporales que puede sufrir una serie temporal sin "*modificar su esencia*". Las deformaciones temporales que se producen en la comparación de dos series (eliminación e inserción) se reducen a realizar extensiones sobre ambas series.

La caracterización fundamental de la distancia DTW, que la hace adecuada al problema de comparación con deformación, es que identifica como iguales dos series temporales que sean deformaciones temporales. El siguiente teorema fundamental justifica el comportamiento de la distancia DTW frente a las series temporales.

Teorema C.19. Caracterización Fundamental. *Sea* $\mathcal{T}$ *el conjunto de las series temporales de dimensión finita pero variable, sean* $\boldsymbol{X}, \boldsymbol{Y} \in \mathcal{T}$, $\boldsymbol{X} = [x_1, \ldots, x_n]$, $\boldsymbol{Y} = [y_1, \ldots, y_m]$.

1. $\boldsymbol{X} \gg \boldsymbol{Y}$ *o* $\boldsymbol{Y} \gg \boldsymbol{X}$ *entonces* $d_{DTW}(\boldsymbol{X}, \boldsymbol{Y}) = 0$.
2. *Si* $d_{DTW}(\boldsymbol{X}, \boldsymbol{Y}) = 0$ *entonces existe* $\boldsymbol{X}^* \in \mathcal{T}$ *tal que* $\boldsymbol{X} \gg \boldsymbol{X}^*$ *e* $\boldsymbol{Y} \gg \boldsymbol{X}^*$.
3. *Si* $\boldsymbol{Y} \gg \boldsymbol{X}$ *entonces* $d_{DTW}(\boldsymbol{Y}, \boldsymbol{Z}) \geq d_{DTW}(\boldsymbol{X}, \boldsymbol{Z})$ *para cualquier serie* $\boldsymbol{Z} \in \mathcal{T}$.
4. *Sean* $\boldsymbol{X}$ *e* $\boldsymbol{Y}$ *tales que* $\boldsymbol{X} \gg \boldsymbol{Z}$ *e* $\boldsymbol{Y} \gg \boldsymbol{Z}$ *entonces* $d_{DTW}(\boldsymbol{X}, \boldsymbol{Y}) = 0$.

Para concluir, se presenta la primera utilidad de la curva de deformación. Sea $\Phi = (\phi_x, \phi_Y)$ la curva de deformación de $\mathbf{X}$ frente a $\mathbf{Y}$, es posible construir dos series temporales $\mathbf{X}'$ e $\mathbf{Y}'$ extensiones de $\mathbf{X}$ e $\mathbf{Y}$ respectivamente tales que

$$d_{DTW}(\mathbf{X}, \mathbf{Y}) = d_{DTW}(\mathbf{X}', \mathbf{Y}') = \sum_{i=1}^{k} d(x_i', y_i') \tag{C.5}$$

construidas como

$$x_i' = x_{\phi_X(i)} \quad e \quad y_i' = y_{\phi_Y(i)}$$

Es más, si $d_{DTW}(\mathbf{X}, \mathbf{Y}) = 0$ si y solo si $\mathbf{X}' = \mathbf{Y}'$.

Ejemplo C.20. Sean $\mathbf{X} = [1, 2, 3]$ e $\mathbf{Y} = [1, 3, 3, 4, 5]$ dos series temporales. El valor de $d_{DTW}(\mathbf{X}, \mathbf{Y})$ es 2 y la curva de deformación es:

$$\Phi(1) = (1, 1), \quad \Phi(2) = (2, 1), \quad \Phi(3) = (3, 2), \quad \Phi(4) = (3, 3), \quad \Phi(5) = (3, 4)$$

Para $k = 1$ se produce una eliminación. Para $k = 2$ se produce una coincidencia. Para $k = 3$ y $k = 4$ se producen dos inserciones. Para las extensiones de las series temporales por Φ, en el instante 2 (eliminación) se duplica y_1 y en los instantes 4 y 5 (inserción) se triplica x_3.

$$\begin{array}{rcccl} \mathbf{X}' & = & [x_1, x_2, x_3, x_3, x_3] & = & [1, 2, 3, 3, 3] \\ \mathbf{Y}' & = & [y_1, y_1, y_2, y_3, y_4] & = & [1, 1, 3, 3, 4] \end{array}$$

C.2.4. Restricciones que debe verificar la matriz DTW

En esta sección se introducen una serie de restricciones que deben verificarse para que el algoritmo de cálculo de la DTW mediante la matriz DTW sea completo y exista solución.

Definición C.21. *Sean $\boldsymbol{X} = [x_1, \ldots, x_n]$ e $\boldsymbol{Y} = [y_1, \ldots, y_n]$ dos series temporales, y sea $A_{DTW} = (a_{i,j})_{1 \le i \le n,\ 1 \le j \le m}$ la matriz de distancias del algoritmo DTW que compara $\boldsymbol{X}$ frente a $\boldsymbol{Y}$.*

1. *Se dice que dos índices $i \in \{1 \ldots n\}$ y $j \in \{1, ..m\}$ son **coincidentes** sí el elemento $a_{i,j}$ está definido ($a_{i,j} < \infty$).*
2. *Sean $i_0 \in \{1 \ldots n\}$ y sea $j_0 \in \{1 \ldots m\}$ índices, se define la **coincidencia óptima** respecto de (i_0, j_0) como el par de índices (i, j) tales que*

$$(i, j) = \begin{cases} \underset{(\alpha,\beta)}{\arg\min}(a_{\alpha,\beta} \ / \ \alpha \in \{i_0, i_0 + 1\} \ y \ \beta \in \{j_0, \ j_0 + 1\}) & \text{sí } i_0 + 1 \le n \ y \ j_0 + 1 \le m \\ (i_0, j_0 + 1) & \text{sí } i_0 = n \ y \ j_0 < m \\ (i_0 + 1, j_0) & \text{sí } i_0 < n \ y \ j_0 = m \end{cases}$$

*En estas condiciones, se dice que j es **coincidencia óptima** de i respecto de (i_0, j_0).*

Para concluir esta sección, se va a justificar que el término $(n, m)-ésimo$ de la matriz A_{DTW} es efectivamente el valor $d_{DTW}(\mathbf{X}, \mathbf{Y})$, Los detalles de las siguientes afirmaciones pueden consultarse en [25].

Por construcción del algoritmo, se verifica que el valor $\min(a_{i,j-1}, a_{i-1,j}, a_{i-1,j-1}) + d(x_i, y_j)$ indica el coste del camino óptimo que une $(1, 1)$ con (i, j) teniendo en cuenta las restricciones temporales. Por tanto, para casa $s = 1 \ldots n$

$$a_{\phi_X(s),\phi_Y(s)} = \sum_{i=1}^{s} d(x_{\phi_X(i)}, y_{\phi_Y(i)}) \tag{C.6}$$

Y como consecuencia inmediata se tiene que

$$d_{DTW}(\mathbf{X}, \mathbf{Y}) = a_{n,m} = \sum_{k=1}^{P} d(x_{\phi_X(k)}, y_{\phi_Y(k)}) \tag{C.7}$$

Lo que justifica que el algoritmo DTW halla efectivamente el valor $d_{DTW}(\mathbf{X}, \mathbf{Y})$.

Como por definición todo camino por deformación debe cumplir unas condiciones temporales, para que el algoritmo DTW sea completo y exista solución dichas restricciones deben traducirse sobre la matriz DTW. Así pues, se tienen que cumplir las siguientes restricciones.

Restricciones supuestas sobre las series temporales y su relación.

Sean $\mathbf{X} = [x_1, \ldots, x_n]$ e $\mathbf{Y} = [y_1, \ldots, y_m]$ dos series temporales con matriz DTW dada por $A_{DTW} = (a_{i,j})_{1 \leq i \leq n, 1 \leq j \leq m}$, para que el algoritmo DTW sea completo (en cualquiera de las versiones que se presenten) y tenga sentido como tal se han supuesto las siguientes hipótesis sobre las series temporales y su relación.

1. Cada índice de la serie muestra debe coincidir con al menos un índice de la serie prueba. Es decir,

   para cada $i \in \{1 \ldots n\}$ existe un $j \in \{1, ..m\}$ tal que $a_{i,j}$ está definido

2. Cada índice de la serie prueba debe coincidir con al menos un índice de la serie muestra. Es decir,

   para cada $j \in \{1 \ldots m\}$ existe un $i \in \{1, ..n\}$ tal que $a_{i,j}$ está definido

   En general, las coincidencias descritas en *1* y *2* se suelen dar en más de un par e índices. En la versión primera del algoritmo DTW descrita esta coincidencia se da para cada índice de la primera serie con todos los índices de la segunda serie y viceversa.

3. El primer índice de la serie muestra debe coincidir con el primer índice de la serie prueba. El último índice de la serie muestra debe coincidir con el último índice de la serie prueba. Esto es:

   $$a_{1,1} < \infty \quad \text{y } a_{n,m} < \infty$$

C.2.5. Teorema de la curva de deformación

En esta sección se recoge el segundo teorema fundamental que justifica el algoritmo para hallar la curva de deformación.

Teorema C.22. *sean* $\boldsymbol{X} = [x_1, \ldots, x_n]$ *e* $\boldsymbol{Y} = [y_1, \ldots, y_m]$ *dos series temporales, el conjunto*

$$\Gamma = \{(i_s, j_s) \ / \ (i_s, j_s) \text{ coincidencia óptima respecto de } (i_{s-1}, j_{s-1}),\ s = 2 \ldots P-1, \} \\ \text{con } (i_1, j_1) = (1,1) \text{ y } (i_P, j_P) = (n, m)$$

es una curva de deformación de la comparación de $\boldsymbol{X}$ *frente a* $\boldsymbol{Y}$.

En ningún momento se afirma que el camino Γ del teorema anterior sea la única curva de deformación. Es más, la curva de deformación puede no ser única (véase el *Ej. C.24*) y depende de algunas consideraciones de la implementación del algoritmo.

El algoritmo que permite el cálculo de la curva de deformación, como consecuencia del teorema anterior viene definido como sigue.

Significado geométrico de la curva de deformación.

Con la notación de la curva de deformación $\Phi(k) = (\phi_X(k), \phi_Y(k))$, según la *Definición C.13* se tienen tres posibles relaciones entre $\Phi(k)$ y $\Phi(k+1)$. Geométricamente, estas relaciones están enlazadas con un "movimiento" sobre la matriz A_{DTW}, y suponen un comportamiento en el transcurso del tiempo entre ambas series.

- Una *deformación en el tiempo por* ***eliminación*** en $k+1$ se produce cuando

  $$\phi_X(k+1) = \phi_X(k) + 1 \quad \text{y} \quad \phi_Y(k+1) = \phi_Y(k)$$

  y representa *movimiento en horizontal* en la matriz A_{DTW} . Esto implica que el tiempo en la secuencia de prueba $\mathbf{Y}$ se "*detuvo*". Una sucesión de movimientos en horizontal supone que la secuencia prueba se desacelero respecto de la secuencia muestra durante un intervalo de tiempo.

- Una *deformación en el tiempo por* ***inserción*** en $k+1$ se produce cuando

  $$\phi_X(k+1) = \phi_X(k) \quad \text{y} \quad \phi_Y(k+1) = \phi_Y(k) + 1$$

  y representa un *movimiento en vertical* en la matriz A_{DTW}. Esto implica que el tiempo de la secuencia muestra $\mathbf{X}$ se "*detuvo*". Una sucesión de movimientos en vertical supone que la secuencia prueba se aceleró respecto de la secuencia muestra durante un intervalo de tiempo.

- Una ***coincidencia temporal*** en $k+1$ entre ambas series se produce cuando

  $$\phi_X(k+1) = \phi_X(k) + 1 \quad \text{y} \quad \phi_Y(k+1) = \phi_Y(k) + 1$$

  y representa un *movimiento en diagonal* en la matriz A_{DTW} . Esto significa que la secuencia muestra $\mathbf{X}$ y la secuencia prueba $\mathbf{Y}$ mantuvieron una misma velocidad.

C.2.6. Algoritmo para determinar las asociaciones de índices

Las asociaciones entre índices de dos series temporales $\mathbf{X} = [x_1, \ldots, x_n]$ e $\mathbf{Y} = [y_1, \ldots, y_m]$ (véase *Definición C.21*) establecen cuando un mismo "evento" sucede en un momento temporal diferente y/o a diferentes velocidades. En el ejemplo introductorio de la *Figura 4.1* el evento "comienza a descender" se produce en el momento 4 de la primera serie y en el 6 de la segunda serie. Esta sección está basado en [7] y [8].

A partir de la curva de deformación se establece y define la relación de asociación entre los índices de la serie muestra con la serie prueba. En la siguiente definición se desarrolla el concepto.

Definición C.23. *Sean $\boldsymbol{X}$ e $\boldsymbol{Y}$ dos series temporales muestra y prueba con conjuntos de índices $\{1 \ldots n\}$ y $\{1 \ldots m\}$ respectivamente, sea $\Phi = (\phi_x, \phi_y)$ la curva de deformación de cardinal P, se dice que un índice $i \in \{1 \ldots n\}$ de la serie muestra se asocia con un índice $j \in \{1 \ldots m\}$ de la serie prueba si existe $k \in \{1 \ldots P\}$ tal que $(i, j) = \Phi(k)$.*

En particular, interesa dado un índice i de la serie muestra conocer los índices j de la serie prueba que se asocian con i. En principio, se pueden dar tres situaciones que dependen de la relación temporal (deformación o coincidencia) en dicho momento.

- Si se ha producido una eliminación entonces existen al menos dos índices de la serie muestra que se asocian con un mismo índice de la serie prueba.
- Si se ha producido una inserción entonces para un índice de la serie muestra existen al menos dos índices de la serie prueba que se asocian con dicho índice.

Sean $\mathbf{X} = [x_1, \ldots, x_n]$ e $\mathbf{Y} = [y_1, \ldots, y_m]$ dos series temporales arbitrarias y sea $\Phi = (\phi_X, \phi_Y)$ la curva de deformación de cardinal P. Se denota por $\gamma(s)$, $s = 1 \ldots n$, al conjunto de los índices que se asocias con s. Es decir,

$$\gamma(s) = \{\phi_Y(t) : \ t \in \{1 \ldots P\} \text{ tales que } \phi_X(t) = s\}$$

O escrito de otra forma

$$\gamma(s) = \{j \in \{1 \ldots m\} \text{ tales que } (s, j) \in \Phi\}$$

C.2.7. Ejemplo DTW

Para concluir, se presenta un ejemplo de cálculo de la DTW, curva de deformación y asociaciones de índices, basado en las series del *Ejemplo C.17*.

Ejemplo C.24. Se consideran las series temporales $\mathbf{X} = [3,3,6,0,3,3]$ e $\mathbf{Y} = [3,3,3,6,0,3]$. Denotamos por $E = (e_{i,j})_{1\leq i,j\leq 6}$ la matriz de distancias para la distancia dada por $d(a,b) = |a-b|$ (es decir, $e_{i,j} = |x_i - y_j|$) y por A_{DTW} la matriz del algoritmo. Aplicando las ecuaciones *(4.7), (4.8) y (4.9)* se tiene que:

$$E = \begin{pmatrix} 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 & 0 \\ 3 & 3 & 3 & 6 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \qquad A_{DTW} = \begin{pmatrix} \mathbf{0} & 0 & 0 & 3 & 6 & 6 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & 3 & 6 & 6 \\ 3 & 3 & 3 & \mathbf{0} & 6 & 9 \\ 6 & 6 & 6 & 6 & \mathbf{0} & 3 \\ 6 & 6 & 6 & 9 & 3 & \mathbf{0} \\ 6 & 6 & 6 & 9 & 6 & \mathbf{0} \end{pmatrix}$$

La curva de deformación (marcada en negrita en la matriz anterior), según el algoritmo anterior, viene dada por:

$$\begin{array}{llll} \Phi(1) = (1,1), & \Phi(2) = (2,1), & \Phi(3) = (2,2), & \Phi(4) = (2,3) \\ \Phi(5) = (3,4), & \Phi(6) = (4,5), & \Phi(7) = (5,6), & \Phi(8) = (6,6) \end{array}$$

En el ejemplo se puede observar como la definición dada para la curva de deformación no implica su unicidad. Es más, es también curva de deformación la definida como:

$$\begin{array}{llll} \Psi(1) = (1,1), & \Psi(2) = (1,2), & \Psi(3) = (1,3), & \Psi(4) = (2,3) \\ \Psi(5) = (3,4), & \Psi(6) = (4,5), & \Psi(7) = (5,6), & \Psi(8) = (6,6) \end{array}$$

La elección de Φ o Ψ como curva de deformación no altera el valor de la distancia DTW y depende de la implementación del algoritmo. En la implementación en pseudocódigo dada en la *Sección B.3*, este matiz se da en los condicionales desde la línea 23 a 30. Para un índice (i,j) en primer lugar se comprueba el coeficiente $a_{i-1,j}$ lo que hace que la curva obtenida sea Φ. Por último, el ejemplo muestra como analíticamente es posible hallar más de una curva de deformación y no necesariamente tiene que haber unicidad.

Las asociaciones de índices son:

$$\begin{array}{lll} \gamma(1) = 1 & \gamma(2) = \{1,2,3\} & \gamma(3) = 4 \\ \gamma(4) = 5 & \gamma(5) = 6 & \gamma(6) = 6 \end{array}$$

Finalmente, se puede observar el significado de las deformaciones temporales en función de las identificaciones de índices. Recuperando la primera curva de deformación del ejemplo anterior:

1. Si se produce una **inserción** en k (caso de $k = 3$, $k = 4$ en Φ) entonces el conjunto $\gamma(2)$ tiene al menos dos elementos (en este caso tres por haberse producido dos inserciones).
2. Si se produce una **eliminación** en k (caso de $k = 2$ o $k = 8$ en Φ) entonces la intersección $\gamma(1) \cap \gamma(2) = \{1\} \neq \varnothing$.

C.2.8. Otras versiones del Algoritmo DTW

Existen diferentes versiones del algoritmo DTW que suponen una mejora respecto de la *versión primera* que se presenta en la *Sección 4.3.3*. Sean $\mathbf{X} = [x_1, \ldots, x_n]$ e $\mathbf{Y} = [y_1, \ldots, y_m]$ dos series temporales cuyos coeficientes quedan definidos en el espacio métrico $(\mathbb{K}, d)$, y sea $A_{DTW} = (a_{i,j})_{i=1..n;j=1..m}$ la matriz resultante del algoritmo *DTW* correspondiente.

1. Cambios en el patrón de recursividad.

El **patrón de recursividad** hace referencia a la *relación recursiva* que permite calcular los coeficientes de la matriz A_{DTW}. En la primera versión del algoritmo (*página 28*) este patrón está definido por *(4.9)*. Para más detalles sobre los patrones de recursividad, y para conocer otro tipo de parones diferentes a los simétricos aquí desarrollados, consúltese [9]

Cambiando ligeramente el problema que define la distancia DTW (véase *(C.3)*) este se puede plantear con pesos. En la versión más general, dadas dos series temporales $\mathbf{X} = [x_1, \ldots, x_n]$ e $\mathbf{Y} = [y_1, \ldots, y_m]$ y dado Ψ un camino de deformación de cardinal P se define la distorsión acumulada para $\Psi = (\psi_X, \psi_Y)$ respecto de $\mathbf{X}$ e $\mathbf{Y}$ como

$$dist_\Psi(\mathbf{X}, \mathbf{Y}) = \sum_{i=1}^{P} d(x_{\psi_X(i)}, y_{\psi_Y(i)}) \cdot m_\Psi(i) M_\psi$$

donde $m_\Psi(i)$ es el peso para el paso $i-ésimo$ y M_Ψ es una constante de normalización que permite comparar la distorsión acumulada de dos series. En este nuevo contexto, la distancia DTW se define como

$$d^*_{DTW}(\mathbf{X}, \mathbf{Y}) = \min_{\Psi \in \mathcal{P}_{n,m}} dist_\Psi(\mathbf{X}, \mathbf{Y})$$

Generalmente, los pesos para el paso suelen ser constantes que no dependen ni de la serie temporal ni del valor de los índices, sino que dependen del tipo de deformación o coincidencia producida (es decir, de la relación entre $\Psi(s)$ y $\Psi(s+1)$ en las tres posibles opciones dadas en la *Def. C.13*). En el segundo patrón, se define

$$m_\Psi(s) = \begin{cases} 1 & \text{si } \Psi(s) \text{ no es coincidencia} \\ 2 & \text{si } \Psi(s) \text{ es coincidencia} \end{cases}$$

Y el valor de normalización es $n + m$.

El cambio propuesto en esta variación viene de sustituir *(4.9)* por

$$a_{i,j} = \min\{a_{i-1,j-1} + \alpha \cdot d(x_i, y_j), a_{i-1,j} + d(x_i, y_j), a_{i,j-1} + d(x_i, y_j)\} \alpha \in \mathbb{R} \tag{C.8}$$

Esta modificación busca "favorecer" que se produzcan deformaciones temporales frente a las coincidencias, penalizando la situación en la cual dos componentes, una de cada serie, son próximas entre sí, pero no iguales.

A los patrones de recursividad dados en *(4.9)* y *(C.8)* se les conoce como **patrones simétricos**, pues su construcción trata por igual al coeficiente $a_{i,j-1}$ y al coeficiente $a_{i-1,j}$.

2. Aplicación de restricciones de localidad.

El principal problema que presentan ambos patrones de recursividad en la versión primera del algoritmo DTW es que permite una **deformación temporal total**. Esta se produce cuando en la curva de deformación se establece la identificación de todos los índices de una de las series temporales respecto al primer o último índice. Esta deformación supone que el total del tiempo de una de las series transcurre mientras el tiempo permanece detenido en la otra serie. En la curva de deformación se observaría que:

$$\Phi = \{(1,1), \ldots, (n,1), (n,2), \ldots, (n,m)\}$$
$$\text{ó}$$
$$\Phi = \{(1,1), \ldots, (1,m), (2,m), \ldots, (n,m)\}$$

La solución a este problema es agregar una *restricción de localidad sobre la deformación en el tiempo*. Existen diferentes restricciones de localidad, desde funciones simples, funciones dependientes del tiempo, etc.

La restricción más sencilla es la **limitación constante**. Se toma un valor $\omega \in \mathbb{N}$ de **tolerancia**, y se impiden deformaciones en el tiempo superiores a ω. Es decir, si $\mathcal{L}$ denota el cardinal del camino de deformación, se considera el subconjunto de $\mathcal{P}_{n\times m}$ dado por:

$$\mathcal{P}^*_{n\times m} := \{\Psi = (\psi_X, \psi_Y) \in \mathcal{P}_{n\times m} \text{ tales que } \forall s = 1 \ldots \mathcal{L}(\Psi) \text{ cumplen } |\psi_X(s) - \psi_Y(s)| \leq \omega\}$$

Y se redefine la distancia DTW como

$$d_{DTW}(\mathbf{X}, \mathbf{Y}) = \min_{\Psi \in \mathcal{P}^*_{n,m}} \left(\sum_{(i,j)\in\Psi} d(x_i, y_j) \right) \tag{C.9}$$

En términos de la matriz del algoritmo DTW, $A_{DTW} = (a_{i,j})_{i,j}$, se aplica la siguiente restricción: $a_{i,j}$ estará definido ($a_{i,j} \neq \infty$) sí y solo sí $|i - j| \leq \omega$. Para ello se modifican las ecuaciones *(4.7)*, *(4.8)* y *(4.9)*.

Algoritmo DTW *versión segunda*

Sea $\mathbf{X} = [x_1, \ldots, x_n]$ la serie muestra e $\mathbf{Y} = [y_1, \ldots, y_m]$ la serie prueba, cuyos coeficientes pertenecen al espacio métrico $(\mathbb{K}, d)$.

1. Dado $\omega \in \mathbb{N}$, se calculan los coeficientes de la matriz $A_{DTW} = (a_{i,j})_{i,j}$

   1.1. Se define el primer elemento de la matriz DTW:

   $$a_{1,1} = d(x_1, y_1) \tag{C.10}$$

   1.2. Se calcula la primera fila y columna de la matriz DTW.
   Para cada $j = 2 \ldots m$ se define

   $$a_{1,j} = \begin{cases} d(x_1, y_j) + a_{1,j-1} & \text{Sí } |i-j| \leq \omega \\ \infty & \text{Sí } |i-j| > \omega \end{cases} \tag{C.11}$$

   Y para cada $i = 2 \ldots m$ se define

   $$a_{i,1} = \begin{cases} d(x_i, y_1) + a_{1,j-1} & \text{Sí } |i-j| \leq \omega \\ \infty & \text{Sí } |i-j| > \omega \end{cases} \tag{C.12}$$

   1.3. Para cada $i = 2 \ldots n$ y para cada $j = 2 \ldots m$ se define el elemento $a_{i,j}$ como:

   $$a_{i,j} = \begin{cases} d(x_i, y_j) + \text{mín}(a_{i,j-1}, a_{i-1,j}, a_{i-1,j-1}) & \text{Sí } |i-j| \leq \omega \\ \infty & \text{Sí } |i-j| > \omega \end{cases} \tag{C.13}$$

2. La distancia entre ambas series es el término $a_{n,m}$

Con la restricción sobre los índices se impiden deformaciones temporales de una longitud superior a ω. En principio, ω puede ser un natural arbitrario. Sin embargo, para que exista una solución y el algoritmo sea completo, además de que se cumplan todas las restricciones sobre las series prueba y muestra expuestas en la *Sección C.2.4*, es necesario que $\omega \geq |n-m|$. Así, esta versión del algoritmo justifica la *Definición C.21* y hace necesarias las restricciones primera y segunda expuestas en la *página 140*, superfluas en la versión primera.

Modificando oportunamente la implementación iterativa dada para el algoritmo DTW en la primera versión se obtiene la siguiente implementación en pseudocódigo para la segunda versión del algoritmo DTW.

Pseudocódigo del algoritmo DTW versión segunda

```
1  real DistanciaDTW (x: vector [1..n], y: vector [1..m], w: int){
2      //Comprobación del requisito sobre omega
3      w := maximo(|n-m|, omega)
4
5      //Inicialización de la matriz a infinito
6      DTW: = matriz [1..n, 1..m]
7      for i := 1 .. n do
8      for j:= 1 .. m do
9      DTW[i, j] := infinito
10     end_for
11     end_for
12
13     // Inicialización primer termino
14     DTW[1,1] := {distancia(x[1], y[1])
15
16     //Llenado primera fila
17     for j := 1..m do
18         DTW[1,j] = distancia(x[1], y[j]) + DTW[1,j-1]
19     end_for
20
21     //Llenado primera columna
22     for i := 1..m do
23         DTW[i,1] = distancia(x[i], y[1]) + DTW[i-1,1]
24     end_for
25
26     //Llenado del total de la matriz
27     for i := 2..n do
28         for j := maximo(2,i-w) .. minimo(1,i+w) do
29             //Calculo de la distancia entre dos observaciones
30             costo := distancia(x[i], y[j])
31             //Calculo del coeficiente de la matriz DTW
32             DTW [i, j] := costo +
33             + minimo(DTW[i-1, j], DTW[i, j-1], DTW[i-1,j-1])
34         end_for
35     end_for
36     return DTW[n, m]}
```

Teorema C.25. Condición para la completitud. *Es condición necesaria y suficiente para que exista una solución a la comparación de dos series temporales de longitud n y m, y dicha solución sea hallada por el algoritmo DTW en la versión segunda que el valor ω de tolerancia verifique que $\omega \geq |m - n|$.*

3. Distancia p-DTW

Dados los vectores $\mathbf{x} = (x_1, ..., x_n)$, $\mathbf{y} = (y_1, ..., y_n) \in \mathbb{R}^n$ se define la distancia p entre ambos vectores como

$$\left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p}.$$

De manera análoga, se puede definir la distancia p–DTW.

Definición C.26. *Sean* $\boldsymbol{X} = [x_1, \ldots, x_n]$ *e* $\boldsymbol{Y} = [y_1, \ldots, y_m]$ *dos series temporales arbitrarias y sea* Ψ *un camino de deformación de orden* $n \times m$ *y cardinal* S*, se define la* ***función de coste de grado*** p *como*

$$C^p_{\boldsymbol{X},\boldsymbol{Y}}(\Psi) = \left(\sum_{(i,j)\in\Psi} d(x_i, y_j)^p\right)^{1/p} = \left(\sum_{i=1}^{S} d(x_{\Psi_X(i)}, y_{\Psi_Y(j)})^p\right)^p \tag{C.14}$$

Y se define la ***distancia*** p***–DTW*** *de dos series temporales queda como:*

$$d_{p-DTW}(\boldsymbol{X}, \boldsymbol{Y}) = \min_{\Psi\in\mathcal{P}_{n,m}} C^p_{\boldsymbol{X},\boldsymbol{Y}}(\Psi) = \left(\min_{\Psi\in\mathcal{P}_{n,m}} \sum_{(i,j)\in\Psi} d(x_i, y_j)^p\right)^{1/p} \tag{C.15}$$

Respecto a los algoritmos presentados basta realizar los cambios oportunos en los patrones de recursividad. La distancia DTW dada en la *Definición C.15* se corresponde con el caso de $p = 1$. Existe una relación fundamental entre la dicha distancia 1–DTW (dada por *(C.3)* y denotada por d_{DTW}) y la distancia $p - DTW$ (denotada por d_{p-DTW}).

Sea d la distancia utilizada para la comparación, dado $p \in \mathbb{N}$, se toma $d^*(a, b) = d(a, b)^p$ y se considera la distancia DTW según la *Definición C.15* (*versión primera*) inducida para d^* denotada por d^*_{DTW}. Entonces para cualquier par de series temporales $\mathbf{X}$ e $\mathbf{Y}$ se cumple que

$$d_{p-DTW}(\mathbf{X}, \mathbf{Y}) = d^*_{DTW}(\mathbf{X}, \mathbf{Y})^{1/p} \tag{C.16}$$

Donde d_{p-DTW} esta inducida por la distancia d según *(C.15)*.

El caso más usual, que es el implementado en la librería de Python `tslearn` utilizada para el caso de estudio, es la distancia 2–DTW. Como consecuencia inmediata de *(C.5)* se verifica la siguiente relación entre la distancia 2–DTW y la distancia euclídea.

$$d_{2-DTW}(\mathbf{X}, \mathbf{Y}) = d_{2-DTW}(\mathbf{X}', \mathbf{Y}') = \sqrt{\sum_{i=1}^{k}(x_i' - y_i')^2} = d_E(\mathbf{X}', \mathbf{Y}')$$

La *versión tercera del algoritmo DTW* registra los cambios necesarios a realizar basados en el resultado anterior.

El algoritmo DTW *(versión tercera, p-DTW)*

Sean $\mathbf{X} = [x_1, \ldots, x_n]$ e $\mathbf{Y} = [y_1, \ldots, y_m]$ dos series temporales cualesquiera de longitud n y m respectivamente. Sea d una distancia sobre espacio en el cual están definidos los valores de x_i e y_i $\forall i$, dado $p \in \mathbb{N}$.

1. Se construye la matriz $A_{DTW} = (a_{i,j})_{i,j}$

   1.1. Se define el primer elemento de la matriz DTW (A_{DTW}) como la distancia entre x_1 e y_1.

   $$a_{1,1} = d(x_1, y_1)^p \tag{C.17}$$

   1.2. Se calcula la primera fila y columna de la matriz DTW.
   Para cada $j = 2 \ldots m$ se define

   $$a_{1,j} = d(x_1, y_j)^p + a_{1,j-1} \tag{C.18}$$

   Y para cada $i = 2 \ldots m$ se define

   $$a_{i,1} = d(x_i, y_1)^p + a_{i-1,1} \tag{C.19}$$

   1.3. Para cada $i = 2 \ldots n$ y para cada $j = 2 \ldots m$ se define el elemento $a_{i,j}$ como:

   $$a_{i,j} = d(x_i, y_j)^p + \text{mín}(a_{i,j-1}, a_{i-1,j}, a_{i-1,j-1}) \tag{C.20}$$

2. La distancia DTW entre ambas series es el término $(a_{n,m})^{1/p}$.

4. Comentario final sobre las distintas variaciones.

En lo que resta, salvo que se especifique lo contrario se considera una distancia *DTW* genérica, pues en cualquiera de sus versiones (siempre que se cumplan las restricciones que la hacen completa) los algoritmos pueden ser aplicados. En particular, la biyectividad de la aplicación $f : [0, \infty) \longrightarrow [0, \infty)$ dada por $f(t) = t^{1/p}$ hace que todos los resultados estudiados para d_{DTW} (*Def. C.15*) sean aplicables a d_{p-DTW}. Además, cuando la restricción de localidad impuestas en *(C.9)* (*versión segunda*) cumplan las restricciones necesarias recogidas en la *página 140* los resultados también son aplicables, pues el requisito necesario y suficiente es la existencia de curva de deformación. Sin embargo, el resultado obtenido, en cuanto a la medida de comparación y misma curva de deformación, no tiene por qué ser el mismo.

Apéndice D

Información complementaria al fundamento teórico sobre *clustering*

D.1. Concepto de *clustering*

El objetivo principal es clasificar diferentes series temporales para encontrar patrones comunes y posibles anomalías. En este capítulo se presenta el algoritmo de clasificación no supervisada (*clustering*) por excelencia, el algoritmo de k-medias, y se estudian las modificaciones necesarias para poder aplicar dicho algoritmo, que inicialmente se desarrolló para conjuntos de datos cuyos miembros tienen la misma dimensión, sobre un conjunto de series temporales donde la longitud (dimensión) de cada serie es finita pero variable entre sus miembros. Este apéndice se corresponde con el *Capítulo tercero* de [25].

D.2. Algoritmos de clasificación (*clustering*)

En esta sección se presenta la definición formal de un sistema de clústeres y otras características inherentes que resultan de interés.

Definición D.1. *Sea $\mathbb{V}$ un espacio vectorial sobre $\mathbb{R}$ de dimensión finita $r \in \mathbb{N}$ y sea $d : \mathbb{V}\times\mathbb{V} \to \mathbb{R}$ tal que para cada $x, y \in \mathbb{V}$*

(a) $d(x, y) \geq 0$

(b) $d(x, y) = d(y, x)$

Dado $\mathcal{D} \subset \mathbb{V}$ un conjunto finito de cardinal $m \in \mathbb{N}$, $\mathcal{D} = \{x_1, ..., x_m\}$, que representan los datos a clasificar, si se denotan por $K_1, .., K_n$ las n agrupaciones.

1. *Se dice que $K_1, ..K_n$ forman un **sistema de clústeres** si la pertenencia a cada K_i es excluyente del resto y su unión es el conjunto total de datos. Esto es:*

$$\mathcal{D} = \bigsqcup_{i=1}^{n} K_i \text{ (unión disjunta).}$$

2. *A cada K_i se le denomina* ***clúster***.

3. *Se define la* ***matriz de pertenencia*** *como $W = (\omega_{i,j})_{1\leq i\leq m;1\leq j\leq n}$ tal que:*

$$\omega_{i,j} = \begin{cases} 1 & si \quad x_i \in K_j \\ 0 & si \quad x_i \notin K_j \end{cases}$$

4. *Sea $F : \mathbb{K} \times \mathbb{K} \times \cdots \times \mathbb{K} \longrightarrow \mathbb{K}$ la función que determina los centroides de cada clúster, entonces, para el clúster K_i se define su* ***centroide*** *denotado por c_i como:*

$$c_i = F(\omega_{1,i} \cdot x_1,\ \omega_{2,1} \cdot x_2,\ \ldots\ , \omega_{m,i} \cdot x_m)$$

5. *Se define la* ***distancia intra-clúster*** *para el clúster $K_t = \{\alpha_1, ..., \alpha_s\}$ como la suma de las distancias entre cada par de elementos del clúster. Esto es:*

$$\sum_{j=1}^{s} \sum_{i=j+1}^{s} d(\alpha_j, \alpha_i) = \sum_{j=1}^{m} \sum_{i=j+1}^{m} \omega_{i,t} \cdot \omega_{j,t} \cdot d(x_j, x_i)$$

6. *Se define la* ***distancia inter-clúster*** *entre K_i y K_j como:*

$$\text{mín}\{d(x, y) :\ x \in K_i,\ y \in K_j\} = d(K_i, K_j)$$

7. *Se define la* ***variación intra-clúster de*** *K_t, o simplemente variación de K_i, y se denota por $S^t(W)$, como la suma de las distancias entre cada elemento del clúster con su centroide. Esto es:*

$$\mathcal{S}^t(W) = \sum_{x \in K_t} d(x, c_t) = \sum_{i=1}^{m} \omega_{i,t} \cdot d(x_i, c_t)$$

8. *Se define la* ***variación total intra-clúster****, y se denota por $\mathcal{S}(W)$ como la suma de las variaciones de cada clúster. Esto es:*

$$\mathcal{S}(W) = \sum_{t=1}^{n} \mathcal{S}^t(W) = \sum_{t=1}^{n} \sum_{i=1}^{m} \omega_{i,t} \cdot d(x_i, c_t)$$

D.3. Algoritmo de k-medias versión de la matriz de pertenencia

La matriz de pertenencia verifica las siguientes propiedades:

1) $\omega_{i,j} \in \{0, 1\}\ \forall i = 1..m,\ \forall j = 1..n.$

2) $\sum_{j=1}^{n} \omega_{i,j} = 1,\ \forall i = 1..m.$

3) $\sum_{i=1}^{m} \sum_{j=1}^{n} \omega_{i,j} = m.$

Además, sea $\mathcal{D} = \{x_1, .., x_m\}$ un conjunto de datos, toda matriz $A = (a_{i,j})_{1 \leq i \leq m; 1 \leq j \leq n}$ de dimensiones $m \times n$ que verifique las propiedades anteriores es matriz de pertenencia. Es decir, es equivalente conocer la asignación de clústeres a conocer la matriz de pertenencia.

En consecuencia, dado que es equivalente el conocer la asociación de clústeres y la matriz de pertenencia, el algoritmo anterior puede enunciarse en términos de dicha matriz. Esta versión presenta una ventaja de sencillez en su implementación, pues contempla conceptos más abstractos tales como "un dato pertenece a un clúster" como un valor concreto de un coeficiente de la matriz de pertenencia.

Algoritmo de k-medias *(versión de matriz de pertenencia)*

Sean k el número de clústeres predefinidos, $c_1, .., c_k$ los centroides asociados a cada clúster y $\mathcal{D} = \{x_1, .., x_m\}$ el conjunto de datos a clasificar.

1. Inicialización: se inicializan aleatoriamente cada centroide.

2. Asignación: sea $W^p = (\omega_{i,j}^p)_{1 \leq i \leq m; 1 \leq j \leq k}$ la matriz de pertenencia en la p-ésima iteración. Se inicializa $W^p = \mathbf{0}$ (matriz nula) y para cada $i \in \{1, .., m\}$ se toma

$$j = \arg\min_{t=1..k} \{d(x_i, c_t)\} \tag{D.1}$$

   Y se actualiza el coeficiente correspondiente de la matriz de pertenencia

$$\omega_{i,j}^p = 1 \tag{D.2}$$

3. Cálculo de centroides: se reajustan los centroides como la media de los datos de cada clúster. Con la matriz de pertenencia, si $c_i = (c_i^1, .., c_i^n)$ y $x_j = (x_j^1, ...x_j^n)$ entonces:

$$c_i^t = \frac{\sum_{j=1}^{m} \omega_{j,i} \cdot x_j^t}{\sum_{j=1}^{m} \omega_{j,i}} \qquad t = 1 \dots n \tag{D.3}$$

4. Evaluación: Se comprueba el criterio de parada.

   a) Matriz de pertenencia permanece constante en una iteración, $W^p = W^{p-1}$.

   b) La variación total intra-clúster alcanza un valor de tolerancia mínimo. Es decir, fijado $T \in \mathbb{R}$ tolerancia, se finaliza si $\mathcal{S}(W^p) < T$

   Si no se cumple el criterio de parada entonces se repiten los pasos 2, 3 y 4

La implementación en pseudocódigo se recoge en el *Apéndice B.6* y se realiza para la versión de la matriz de pertenencia del algoritmo.

Ejemplo D.2. Una de los conjuntos de datos más conocidos en el contexto de la búsqueda de patrones es la base de datos de los diferentes tipos de flor de iris. En ella se recogen cuatro atributos de tres tipos de flores (iris-Setosa, iris-Versicolor e iris-Virgínica) de las cuales se van a considerar dos:la longitud del pétalo y el ancho del pétalo. La siguiente figura (*Fig. D.1*) ilustra los datos y el resultado de aplicar el algoritmo de k-medias con 3 clústeres.

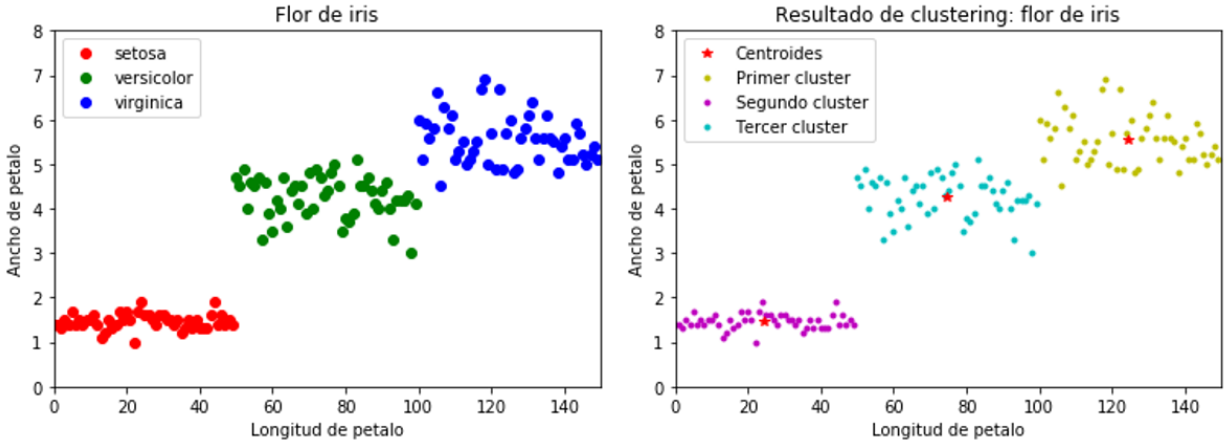


Figura D.1: Ejemplo de *clustering* con la flor de iris: representación de los datos (izquierda) y resultado del algoritmo (derecha)

D.4. Media de Fréchet y algoritmo DBA

Dado que se comparan series temporales de diferente longitud, la media de las series como media aritmética cada componente carece de sentido. En esta sección se busca definir una "media" aplicable a series temporales, para lo que se recurre a la *media de Fréchet*.

Manteniendo la notación, $\mathcal{T}$ denota al conjunto de las series temporales de longitud finita y variable menor que n y d_{DTW} denota la distancia DTW (en cualquiera de sus versiones).

Definición D.3. *Sea* $\mathcal{D} = \{\boldsymbol{X}_1, ..., \boldsymbol{X}_m\} \subset \mathcal{T}$.

1. *Se define la* ***función de Fréchet*** *como la aplicación* $F : \mathcal{T} \to \mathbb{R}$ *tal que para cada* $\boldsymbol{Z} \in \mathcal{T}$

$$F_{\mathcal{D}}(\boldsymbol{Z}) = \sum_{i=1}^{m} d_{DTW}(\boldsymbol{Z}, \boldsymbol{X}_i)$$

2. *Se define la* ***media de Fréchet*** *como la serie temporal que hace mínimo el valor de la función de Fréchet. Esto es:*

$$\mu = \underset{\boldsymbol{M} \in \mathcal{T}}{\arg\min}\ F_{\mathcal{D}}(\boldsymbol{M}) \tag{D.4}$$

La siguiente figura ilustra (*Fig. D.2*) un ejemplo de aplicación de media de Fréchet para promediar un conjunto de series temporales. Para ello se ha utilizado el algoritmo DBA que se presenta posteriormente.

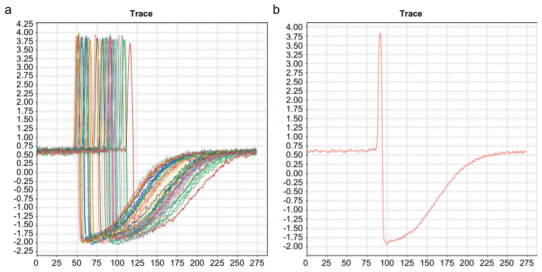


Figura D.2: Representación de las series temporales a promediar *(a)* y del promedio obtenido mediante el algoritmo DBA *(b)*.

Ejemplo D.4. Consideramos las series temporales dadas por $\mathbf{Y} = [4, 4, 7, 4, 1]$, $\mathbf{Z} = [5, 8, 8, 5, 5, 2]$ y Dada $\mathbf{X} = [3, 6, 3, 0]$. Sean Φ y Ψ las curvas de deformación de las comparaciones de $\mathbf{X}$ frente a $\mathbf{Y}$ y $\mathbf{Z}$ respectivamente y sean $\gamma_Y(s)$ y $\gamma_Z(s)$ el conjunto de índices de las series $\mathbf{Y}$ y $\mathbf{Z}$ asociados al índice s de la serie $\mathbf{X}$. Entonces:

$$\begin{array}{lllll} \Phi(1) = (1,1) & \Phi(2) = (1,2) & \Phi(3) = (2,3) & \Phi(4) = (3,4) & \Phi(5) = (4,5) \\ \Psi(1) = (1,1) & \Psi(2) = (2,2) & \Psi(3) = (2,3) & \Psi(4) = (2,4) & \Psi(5) = (3,5) \quad \Psi(6) = (4,6) \\ \gamma_Y(1) = \{1,2\} & \gamma_Y(2) = \{3\} & \gamma_Y(3) = \{4\} & \gamma_Y(4) = \{5\} & \\ \gamma_Z(1) = \{1\} & \gamma_Z(2) = \{2,3,4\} & \gamma_Z(3) = \{5\} & \gamma_Z(4) = \{6\} & \end{array}$$

La media de Fréchet para las series $\mathbf{Y}$, $\mathbf{Z}$ frente a $\mathbf{X}$ es $\mathbf{C}$ dada por:

$$c_1 = \frac{y_1 + y_2 + z_1}{3} = \frac{13}{3} \quad c_2 = \frac{y_3 + z_2 + z_3 + z_4}{4} = 7$$

$$c_3 = \frac{y_4 + z_5}{2} = 4{,}5 \qquad c_4 = \frac{y_5 + z_6}{2} = 1{,}5$$

Dado un conjunto de series temporales, su media de Fréchet (sin especificar respecto a qué serie) se halla generando aleatoriamente series de comparación y repitiendo el procedimiento anterior se selecciona la media de menor distancia DTW. En la generación aleatoria caben estudiar dos atributos de cada serie de comparación:

- El tamaño de la serie
- La generación de las componentes

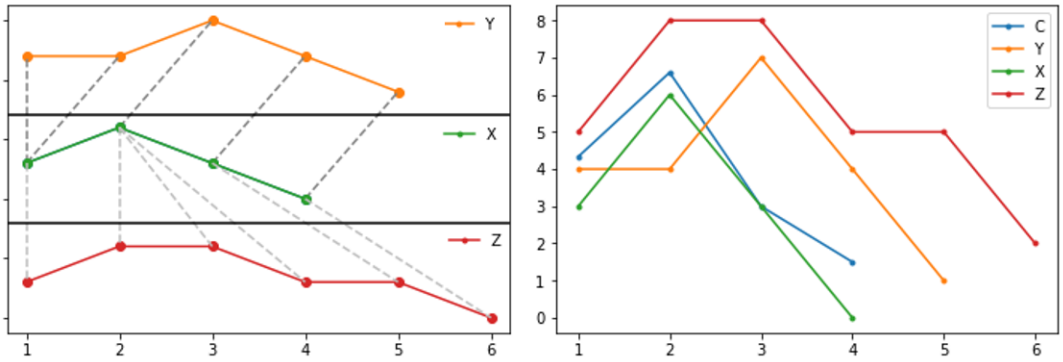


Figura D.3: Representación de las series temporales **X**, **Y** y **Z** con sus asociaciones de índices respecto de **X** (izquierda) y junto a su media **C** (derecha)

En cuanto a la primera de ellas, se suele tomar el máximo de las longitudes de las series temporales que se quieren promediar. Respecto a los valores de las componentes se recurre a técnicas heurísticas.

El algoritmo DBA como solución a la media de Fréchet.

El algoritmo DBA como se ha presentado no resuelve exactamente el problema *(D.4)*, sino que resuelve una variante en la que las series candidatas a promedio tienen longitud fija. Es decir, sea $\mathcal{T}_T \subset \mathcal{T}$ el conjunto de las series temporales de longitud exactamente T, el algoritmo DBA encuentra la solución al problema

$$\mu = \underset{\mathbf{M} \in \mathcal{T}_T}{\arg\min}\ F_{\mathcal{D}}(\mathbf{M}) \tag{D.5}$$

Sin embargo, el algoritmo dado es perfecto para el clustering de series temporales al cumplir las condiciones específicas.

D.4.1. Construcción del algoritmo de k-medias aplicable a series temporales

En esta sección, se recogen las modificaciones pertinentes sobre el algoritmo de k-medias general de la *Sección 4.2.2* para ser aplicable a series temporales de diferente longitud con misma escala temporal (es decir, el conjunto de índices de mayor cardinal es finito y contiene al resto).

Como se permiten series de diferente longitud, la medida de comparación a usar es la distancia DTW (en cualquiera de sus versiones) y los centroides se calculan mediante el algoritmo DBA.

Para su implementación, dado el elevado coste computacional de la distancia DTW (cuya complejidad algorítmica es de $\mathcal{O}(n^2)$), se debe minimizar el número de veces que se calcula dicha distancia. Por ello, en la implementación presentada en el *Apéndice B.7*, se "guarda" el valor de

la distancia de cada dato a su centroide (en el vector `distancias`) necesaria para el cálculo de la variación. Análogamente, se procede al cálculo de las asociaciones de índices de cada centroide respecto de cada dato asignado a su mismo clúster (implementado en la lista `gammas`) aprovechando el momento de cálculo de la matriz DTW correspondiente, y que resultan necesarias para el cálculo de los centroides según el algoritmo DBA. Por último, cabe destacar que los centroides deben ser de la misma longitud y que esta se mantiene en cada iteración.

Algoritmo de k-medias *(versión aplicable a series temporales)*

Sean k el número de clústeres predefinidos, $\mathbf{C}_1, .., \mathbf{C}_k$ los centroides asociados a cada clúster $K_1, ...K_k$ y $\mathcal{D} = \{\mathbf{X}_1, .., \mathbf{X}_m\}$ el conjunto de datos a clasificar. Sea n_i la longitud de la serie $\mathbf{X}_i = [x_1^i, \ldots, x_{n_i}^i]$

1. Inicialización: Se inicializan aleatoriamente cada centroide como series temporales de longitud T. Generalmente, se toma $T = \text{máx}\{n_1, ..., n_m\}$

2. Asignación: sea $W^p = (\omega_{i,j}^p)_{1 \leq i \leq m; 1 \leq j \leq k}$ la matriz de pertenencia en la p-ésima iteración. Se inicializa $W^p = \mathbf{0}$ (matriz nula) y para cada $i \in \{1, .., m\}$ se toma

$$j = \underset{t=1..k}{\arg\min}\{d_{DTW}(x_i, c_t)\} \tag{D.6}$$

   Y se actualiza el coeficiente correspondiente de la matriz de pertenencia

$$\omega_{i,j}^p = 1 \tag{D.7}$$

3. Cálculo de la función de identificación: realizada la asignación de X_i al clúster j – *ésimo*, se calcula γ_i función de identificación de índices de $\mathbf{C}_j$ respecto de $\mathbf{X}_i$.

4. Cálculo de centroides: se reajustan los centroides Se define $\mathbf{C}_j$ como la media de Fréchet (*algoritmo DBA*) sobre el conjunto de las series temporales asociadas al clúster j

$$\{X_i : i = 1..m,\ \omega_{i,j}^p = 1\}$$

5. Evaluación: Se comprueba el criterio de parada.

   a) La matriz de pertenencia permanece constante en una iteración. Es decir, $W^p = W^{p-1}$.

   b) La variación total intra-clúster, definida como $\sum_{t=1}^{n} \sum_{i=1}^{m} \omega_{i,t} \cdot d_{DTW}(\mathbf{X}_i, \mathbf{C}_t)$ alcanza un valor de tolerancia mínimo.

   Si no se cumple el criterio de parada entonces se repiten los pasos 2, 3, 4 y 5

El algoritmo anterior está basado en la versión general con las oportunas modificaciones, tales como cambiar la definición de la media o tomar la distancia DTW como medida de comparación. De forma análoga, en las implementaciones presentadas en el *Apéndice B.7* se han cambiado los métodos `calculo_matriz_pertenencia`, `calculo_variación` y `calculo_centroides` para aplicarse a series temporales, manteniendo (en esencia) la función principal `kmeans`.

En la siguiente sección se presentan los principales problemas y dificultades asociadas al algoritmo de clasificación por k-medias de series temporales. Estas dificultades pueden ser consecuencia del propio algoritmo y/o de su aplicación sobre series temporales.

En el contexto de análisis de datos se vienen realizando numerosos estudios para aplicar la clasificación de clústeres de forma efectiva y eficiente en grandes conjuntos de información. En la actualidad, las líneas de investigación se centran en la escalabilidad, en la efectividad para agrupar formas y datos complejos y en los métodos dirigidos a gestionar datos mixtos o no numéricos.

D.5. Problemas del algoritmo de k-medias

1. **Escalabilidad.** El algoritmo de k-medias en general funciona bien (respecto al tiempo) sobre conjunto de datos pequeños, pero al tratarse de un problema *np-hard*, el tiempo requerido cuando se aumenta el conjunto de datos crece enormemente.
   Es más, aplicado a series temporales con la distancia DTW, supuestos k centroides de longitud T y m series temporales de longitud máxima n, para tan solo una única iteración del algoritmo se tiene una complejidad de $\mathcal{O}(h \cdot T \cdot n \cdot m)$. Esto se debe a que para cada serie temporal se calcula su distancia DTW a cada centroide cuya complejidad es $\mathcal{O}(T \cdot n)$. Además, se requieren varias iteraciones hasta llegar a la condición de parada dada.

2. **Capacidad para tratar con diferentes tipos de datos.** El trabajar con series temporales de diferente longitud implica un problema, pues a priori carecen de sentido operaciones matemáticas básicas como la suma y el producto. Esta misma dificultad se puede extender a otro tipo de datos no numéricos en la versión genérica del algoritmo de clustering.

3. **Descubrir clústeres con formas arbitrarias**. Dependiendo de la distancia usada se predefine la forma de los clústeres. Por ejemplo, en el plano real, la distancia euclídea fuerza a que los clústeres tengan forma de bolas mientras que la distancia del supremo fuerza a que tengan forma de cuadrados. Un caso similar se produce entre las diferentes distancias p–DTW al restringirse a espacios de dimensión n y series de dicha longitud. En la *Figura D.4* se puede observar geométricamente como cualquier bola que se considere es incapaz de separar ambos conjuntos de datos.

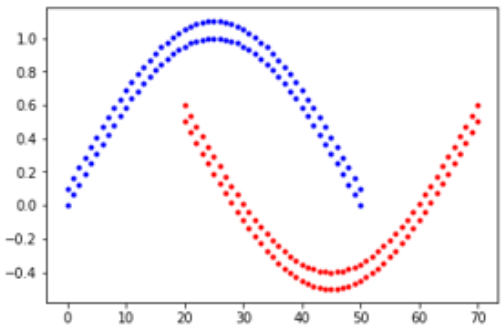


Figura D.4: Ejemplo de agrupaciones complejas, cuya solución falla por k-medias.

4. **Parámetros iniciales predefinidos.** El algoritmo de k-medias requiere que se determinen previamente el número de clústeres y los centroides iniciales para la primera iteración. En particular, una mala elección de los centroides iniciales puede suponer un importante error en la clasificación obtenida.

   En la siguiente figura (*D.5*) se puede observar los efectos sobre el ejemplo de la flor de iris (véase *Ej: D.2*) de elegir erróneamente tanto el número de clústeres como los centroides iniciales. En el primer caso se observa como la clasificación obtenida difiere de la real al agrupar dos tipos de flores distintos en un mismo clúster, pues solo se permitieron dos agrupaciones diferentes. En el caso de inicializar erróneamente los centroides esto ha provocado que se divida en dos el grupo de iris-setosa y se agrupen juntas las variedades iris-versicolor e iris-virginica.

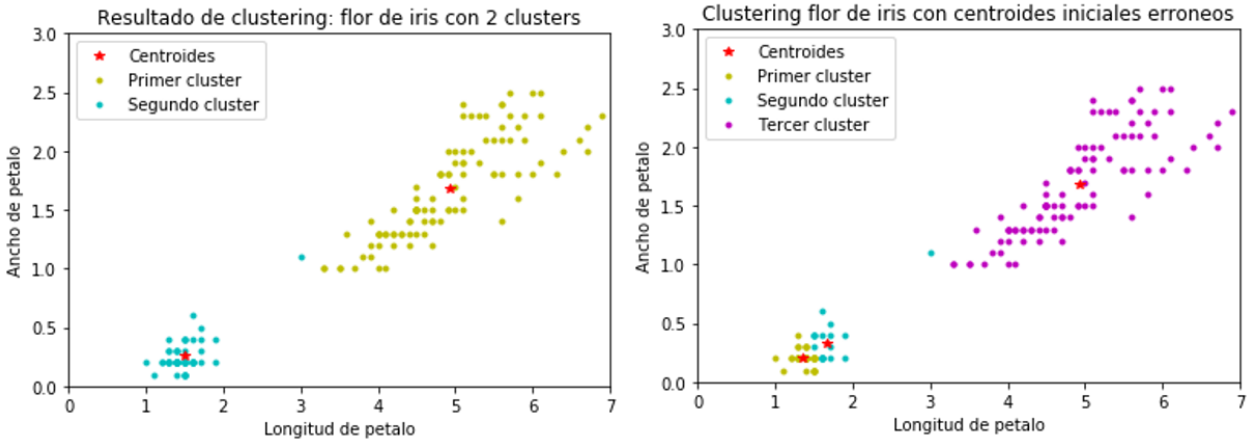


Figura D.5: Ejemplo de clustering incorrectos con la flor de iris: agrupación con dos clústeres (izquierda) y agrupación con los clústeres iniciales erróneos (derecha)

5. **Incapacidad frente a datos ruidosos.** Al igual que la media de un conjunto de datos se ve especialmente afectada por los valores anómalos o *outliers*. El algoritmo de k-medias sufre el mismo problema, pues los *outliers* alejan al centroide del "centro real" del conjunto. Para solventar este inconveniente existe una versión denominada *k-medias recortadas*(*trimmed k-means*). Para más información consúltese [20].

6. **Alta dimensionalidad.** En un conjunto de datos con gran cantidad de atributos se puede sobrecargar el modelo. En primer lugar, puede haber atributos que no sean apropiados para la clasificación (no informativos) y en segundo lugar exigir el uso de demasiados atributos pueden hacer la clasificación demasiado exigente y dar peores resultados además de tener un coste superior.

Apéndice E

Ejemplo de realizaciones de ciertas actividades

En este apéndice se recogen la representación de algunas actividades (operaciones y/o arranques) realizadas (mediante la correspondiente serie temporal dada por las mediciones del sensor de corriente de fase 1), con objeto de ilustrar el caso de estudio y su enorme variabilidad.

También se incluye la evaluación de dichas actividades según su modelo correspondiente para determinar si son usuales o anómalas (recogidas en la *Tabla E.1*). Las actividades representadas han sido elegidas para ilustrar las diferentes alternativas que pueden surgir, tanto usuales como anómalas.

Id	Resultado	Id	Resultado	Id	Resultado
1	Anómalo	2	Usual	3	Usual
4	Usual	5	Anómalo	6	Anómalo
7	Usual	8	Usual	9	Usual
10	Usual	11	Anómalo	12	Usual
13	Usual	14	Usual	15	Usual
16	Usual	17	Anómalo	18	Anómalo
19	Usual	20	Anómalo	21	Usual
22	Usual	23	Usual	24	Anómalo
25	Usual	26	Usual	27	Usual
28	Usual	29	Anómalo	30	Anómalo

Tabla E.1: Resultados de evaluar las actividades seleccionadas como ejemplo por su correspondiente modelo según los identificadores dados en las *Figuras E.1, E.2, E.3, E.4 y E.5*

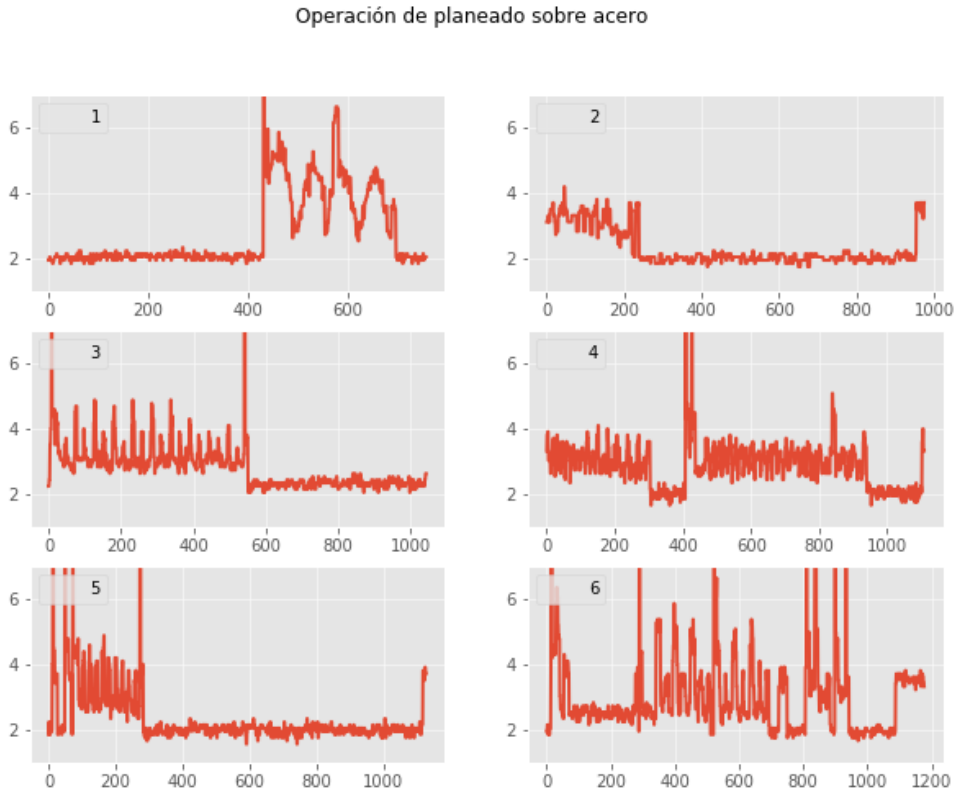


Figura E.1: Representación de las mediciones realizadas por el sensor de corriente de fase 1 para la operación tipo planeado sobre el acero para algunas realizaciones concretas y resultado de clasificar dichas realizaciones por el modelo correspondiente

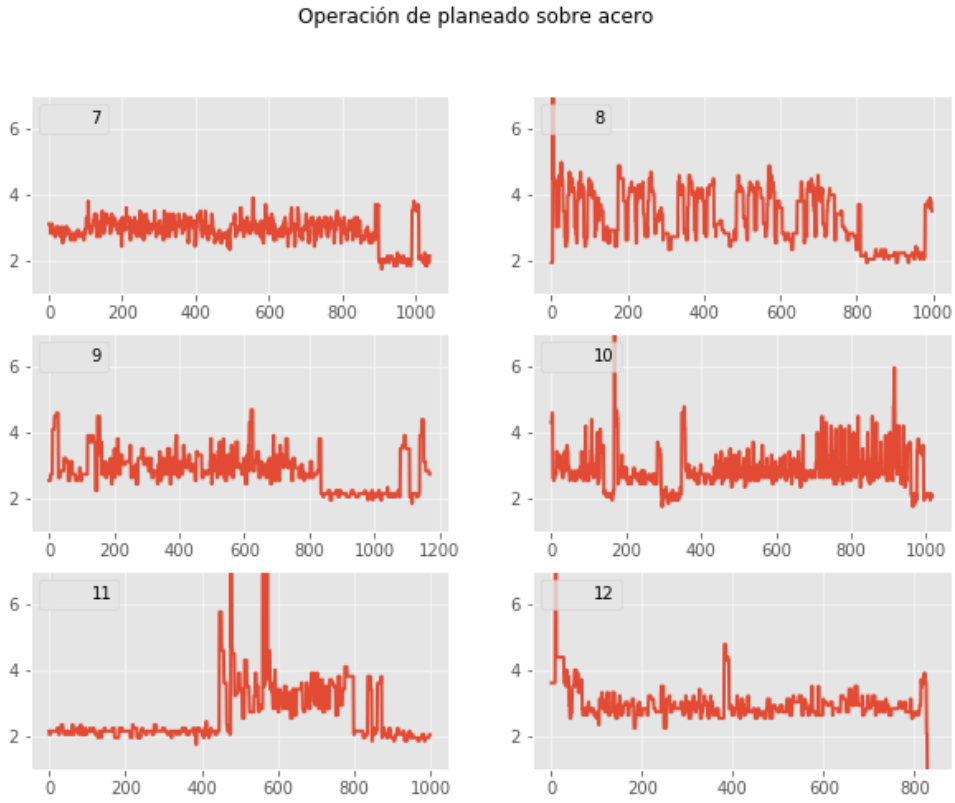


Figura E.2: Representación de las mediciones realizadas por el sensor de corriente de fase 1 para la operación tipo contorneado sobre el acero para algunas realizaciones concretas y resultado de clasificar dichas realizaciones por el modelo correspondiente

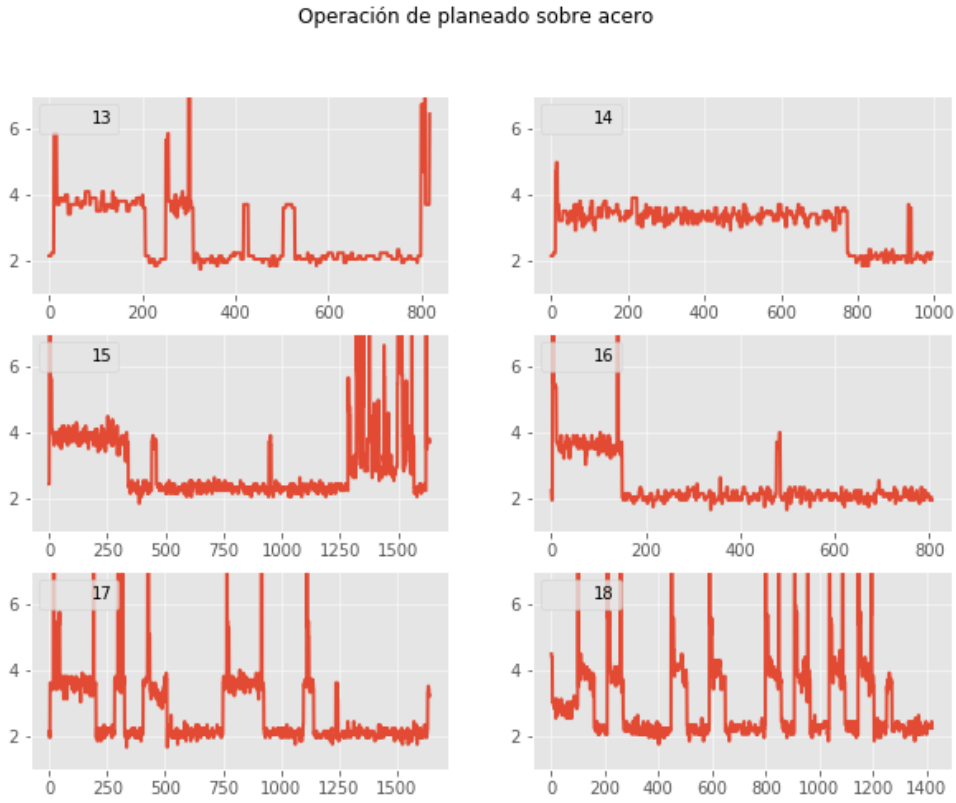


Figura E.3: Representación de las mediciones realizadas por el sensor de corriente de fase 1 para la operación tipo fresado sobre el acero para algunas realizaciones concretas y resultado de clasificar dichas realizaciones por el modelo correspondiente

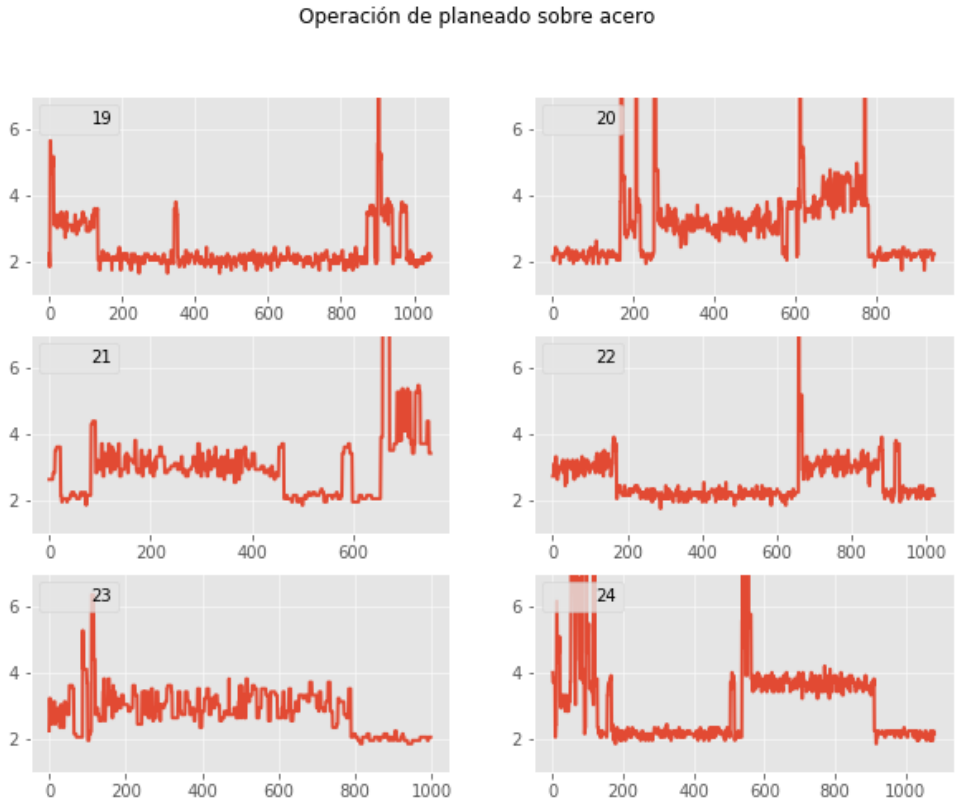


Figura E.4: Representación de las mediciones realizadas por el sensor de corriente de fase 1 para la operación tipo taladrado sobre el acero para algunas realizaciones concretas y resultado de clasificar dichas realizaciones por el modelo correspondiente

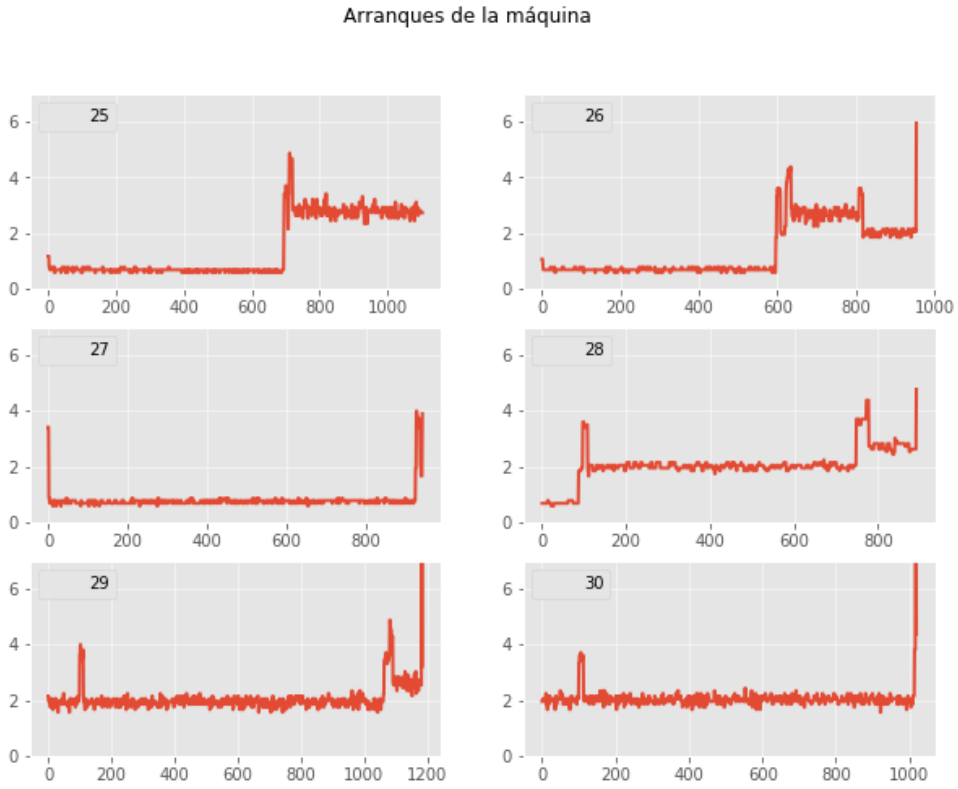


Figura E.5: Representación de las mediciones realizadas por el sensor de corriente de fase 1 para lel arranque de la máquina fresadora y resultado de clasificar dichas realizaciones por el modelo correspondiente

Apéndice F

Resultados del estudio experimental

En este apéndice se recogen los resultados del estudio experimental sobre el resto de operaciones realizadas, cuyo análisis es análogo al realizado en la *Sección 5.3.1* para la operación de contorneado sobre acero.

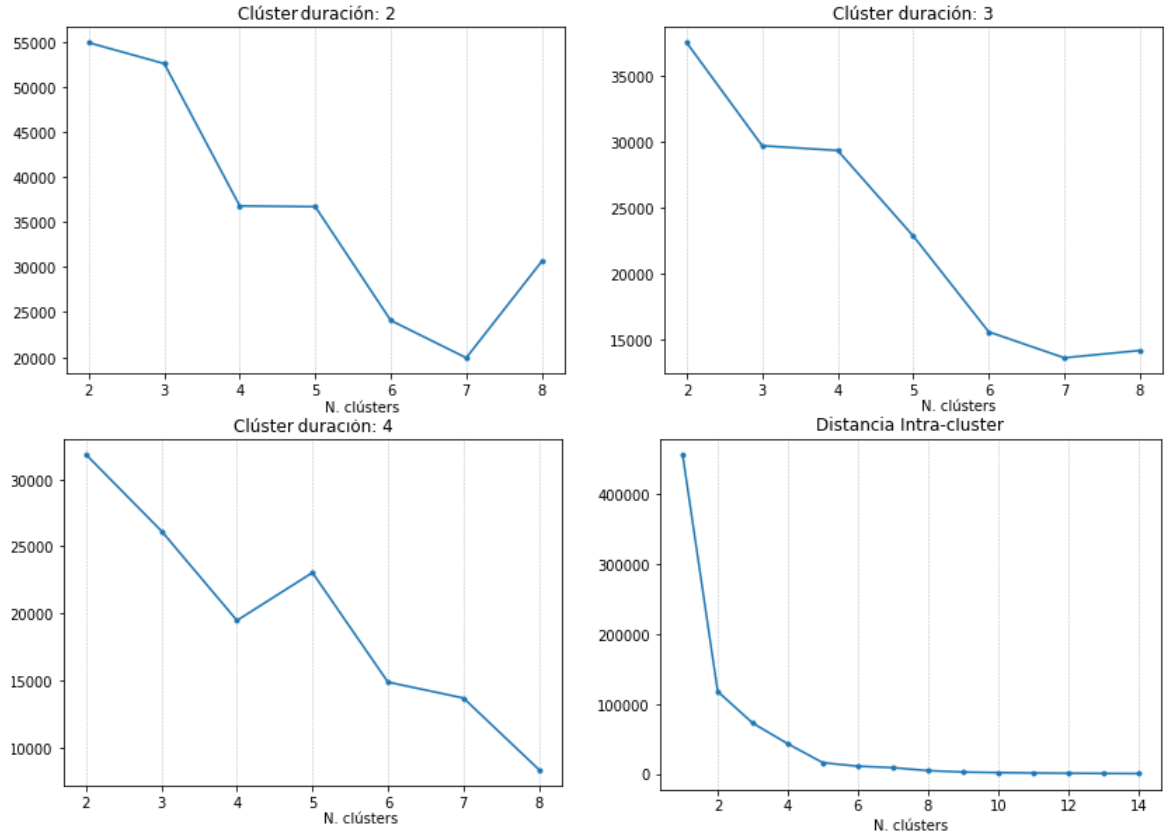


Figura F.1: Representación de la distancia intra-clúster bajo la DTW como función objetivo para la operación de taladrado sobre acero

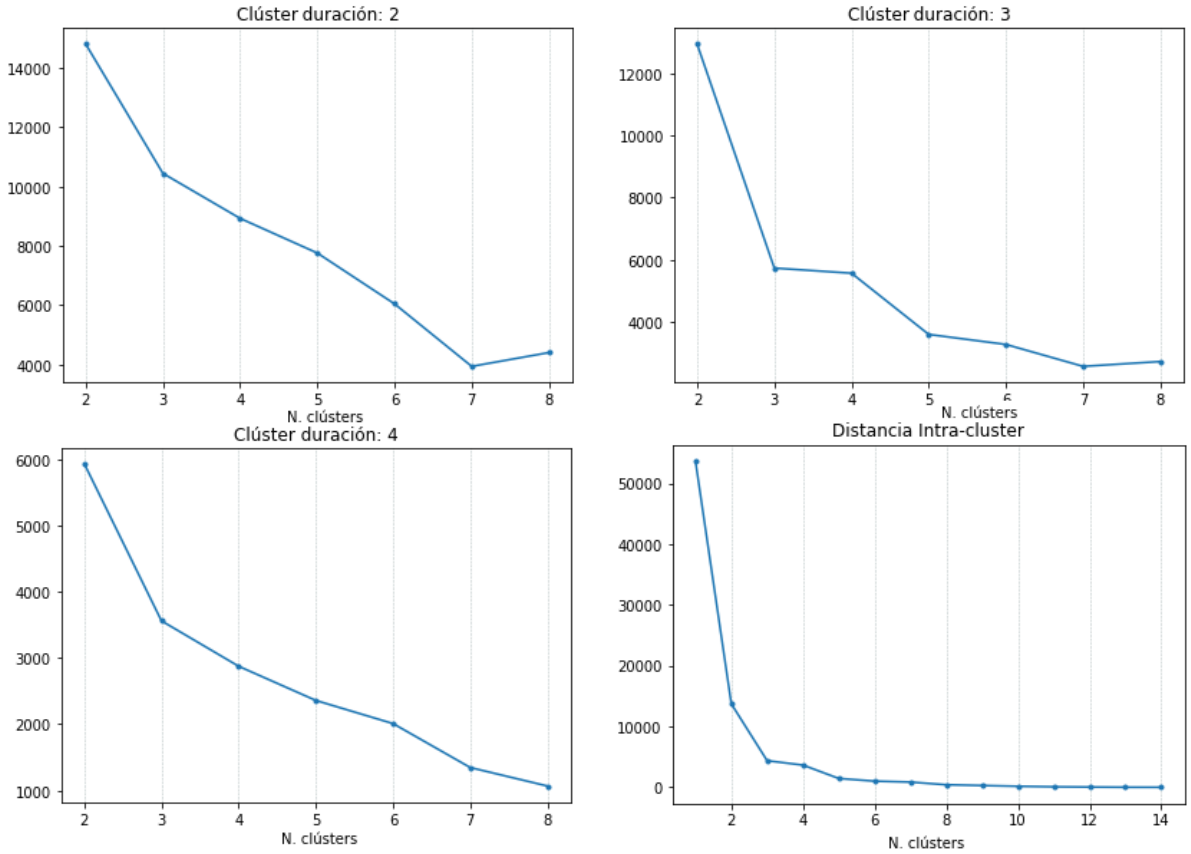


Figura F.2: Representación de la distancia intra-clúster bajo la DTW como función objetivo para la operación de fresado sobre acero.

El estudio incluido en esta sección es común a [25].

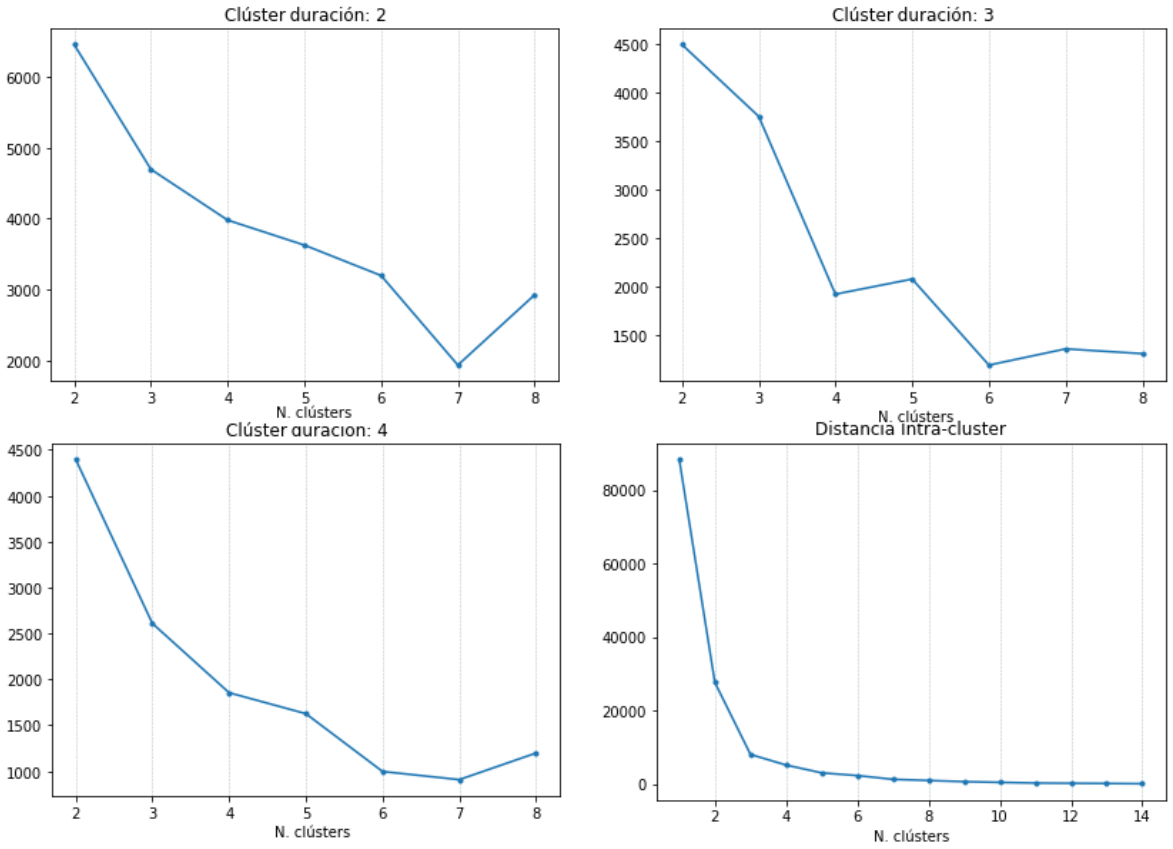


Figura F.3: Representación de la distancia intra-clúster bajo la DTW como función objetivo para la operación de planeado sobre acero

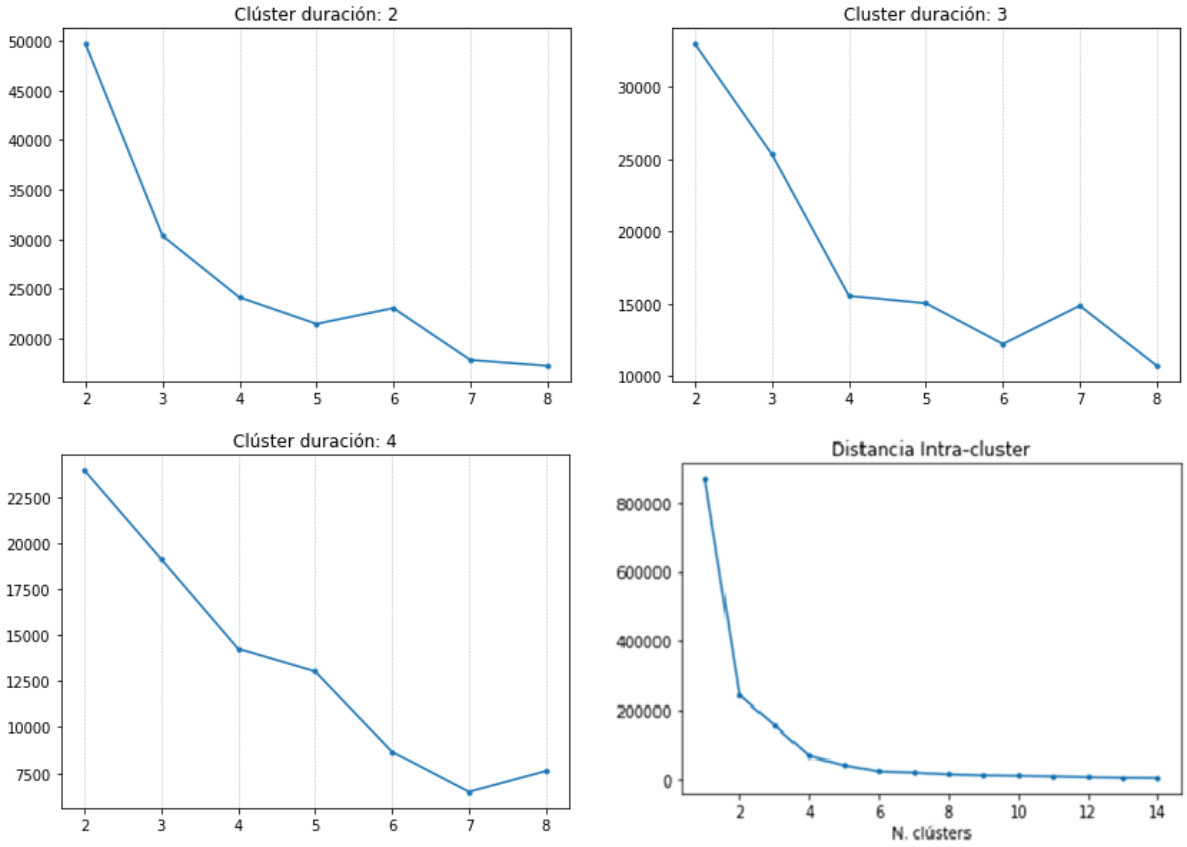


Figura F.4: Representación de la distancia intra-clúster bajo la DTW como función objetivo para para la operación de contorneado sobre acero

Bibliografía

[1] tselarn Documentation, Release 0.3.0

[2] Peter J. Brockwell y Richard A. Davis *Introduction to Time Series and Forecasting, Third Edition*. Ed. Springer.

[3] Geoffrey Grimmett y David Stirzaker *Probability and Random Processes. Third Edition*. Ed. Oxford.

[4] Jacquie Nesbitt *Suavización 1: Métodos Distintos de Suavizado*. Numxl Soporte, Notas Técnicas. Disponible aquí (Mayo 2020)

[5] D. Guijo-Rubio, A. M. Durán-Rosal, P. A. Gutiérrez, A. Troncoso and C. Hervás-Martínez. *Time-Series Clustering Based on the Characterization of Segment Typologies*. IEEE Transactions on Cybernetics, ISSN 2168-2275. Disponible aquí (Junio 2020)

[6] Brijnesh J.Jain. *Making the dynamic time warping distance warping-invariant*. Pattern Recognition, Octubre 2019, Vol. 94, pp. 35-52. Disponible aquí (Junio 2020)

[7] Toni Giorgino. *Computing and Visualizing Dynamic Time Warping Alignments in R: The dtw Package*. Journal of Statistical Software August 2009, Volume 31, Issue 7

[8] Diego F. Silva and Gustavo E. A. P. A. Batista. *Speeding Up All-Pairwise Dynamic Time Warping Matrix Calculation*. SIAM International Conference on Data Mining, May 2016

[9] Z. Zhang, R. Tavenard, A. Bailly, X. Tang, P. Tang, T. Corpetti. *Dynamic time warping under limited warping path length*. Information Sciences, vol. 393, pp. 91–107, 2017

[10] Z. Geler, V. Kurbalija, M. Ivanović, M. Radovanović and W. Dai. *Dynamic Time Warping: Itakura vs Sakoe-Chiba*. 2019 IEEE International Symposium on INnovations in Intelligent SysTems and Applications (INISTA), Sofia, Bulgaria, 2019, pp. 1-6.

[11] F. Petitjean, A. Ketterlin y P. Gancarski. *A global averaging method for dynamic time warping, with applications to clustering*. Pattern Recognition, Elsevier, 2011, Vol. 44, Num. 3, pp. 678-693. Disponible aquí (Junio 2020)

[12] Brijnesh Jain and David Schultz. *On the Existence of a Sample Mean in Dynamic Time Warping Spaces*. arXiv:1610.04460v3 5 Mar 2018. Disponible aquí (Junio 2020)

[13] K. Krishna and M. Narasimha Murty. *Genetic K-Means Algorithm.* IEEE transactions on systems, man, and cybernetics-part B, Vol. 29, N. 3, pp 433-439, junio 1999. Disponible aquí (Junio 2020)

[14] Saeed Aghabozorgi, Ali Seyed Shirkhorshidi y Teh Ying Wah. *Time-series clustering – A decade review.* Elsevier, Vol. 53, pp16-38, octubre-noviembre 2015. Disponible aquí (Junio 2020)

[15] Meena Mahajan, Prajakta Nimbhorkar y Kasturi Varadarajan. *The Planar k-Means Problem is NP-Hard.* Springer, WALCOM 2009: Proceedings of the 3rd International Workshop on Algorithms and Computation, February 2009 pp 274–285. Disponible aquí (Junio 2020)

[16] Álvaro García García *Desarrollo de modelos predictivos adaptados a los nuevos ecosistemas digitales de mantenimiento industrial.* Revista Preditécnico, Núm. 24, pp. 12-15. Disponible aquí (Abril 2020)

[17] Álvaro García García *iDigital4: Nuevos ecosistemas de fabricación basados en el concepto de gemelo digital.* Automática e Instrumentación, Febrero 2020, Num. 517, pp. 38-39. Disponible aquí (Junio 2020)

[18] Hermann, T. Pentek, and B. Otto. *Design Principles for Industrie 4.0 Scenarios: A Literature Review Working Paper A Literature Review*, 2015 (publicado en 2016, 49th Hawaii International Conference on System Sciences (HICSS)). Disponible aquí (Junio 2020)

[19] Klaus Schwab. 2017. *The Fourth Industrial Revolution.* Crown Publishing Group, New York, NY, USA.

[20] Luis A. García-Escudero, Alfonso Gordaliza, Carlos Matrán y Agustin Mayo-Iscar, *A Genera Trimming Approach to Robust Cluster Analysis*, The Annals of Statistics 2008, Vol. 36, Núm. 3, pp 1324–1345. Disponible aquí (Junio 2020)

[21] R. Y. Zhong, X. Xu, E. Klotz, and S. T. Newman, *Intelligent Manufacturing in the Context of Industry 4.0: A Review* Engineering, 2017, Vol. 3, Num. 5, pp. 616–630.

[22] Green. *Econometric Analysis*, 5th ed., Pearson, 2003

[23] Object Managment Group, *OMG Unified Modeling Language (OMG UML), Version 2.5.1*, diciembre 2017. Disponible aquí (Julio 2020)

[24] UWE – UML-based Web Engineering LMU – Ludwig-Maximilians-Universität München Institute for InformaticDisponible aquí (Julio 2020)

[25] Adrián Manzano Santos, *Trabajo de Fin de Grado de Matemáticas: Técnicas de agrupación sobre series temporales para la detección de anomalías en un entorno industrial.* 2020