



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería
Informática**

SmartBeds

**Aplicación de técnicas de
minería de datos para la
detección de crisis epilépticas
y aplicación web
Anexos**



Presentado por José Luis Garrido Labrador
en Universidad de Burgos – 25 de junio
de 2019

Tutores: Dr. Álvar Arnaiz González
y Dr. José Francisco Díez Pastor

Índice general

Índice general	I
Índice de figuras	III
Índice de tablas	v
Apéndice A Plan de Proyecto Software	1
A.1. Introducción	1
A.2. Planificación temporal	1
A.3. Estudio de viabilidad	10
Apéndice B Especificación de Requisitos	13
B.1. Introducción	13
B.2. Objetivos generales	13
B.3. Catálogo de requisitos	13
B.4. Especificación de requisitos	15
Apéndice C Especificación de diseño	27
C.1. Introducción	27
C.2. Diseño de datos	27
C.3. Diseño procedimental	28
C.4. Diseño arquitectónico	30
C.5. Diseño de interfaces	32
Apéndice D Documentación técnica de programación	35
D.1. Introducción	35
D.2. Estructura de directorios	35

D.3. Manual del programador	37
D.4. Compilación, instalación y ejecución del proyecto	40
D.5. Pruebas del sistema	44
Apéndice E Documentación de usuario	47
E.1. Introducción	47
E.2. Requisitos de usuarios	47
E.3. Instalación	47
E.4. Manual del usuario	47
Bibliografía	59

Índice de figuras

B.1. Diagrama casos de uso - Nivel 0	16
B.2. Diagrama CU-2 - Nivel 1	16
B.3. Diagrama CU-3 - Nivel 1	17
B.4. Diagrama CU-4 - Nivel 1	17
C.1. Diagrama entidad-relación	27
C.2. Diagrama relacional	28
C.3. Diagrama de secuencia de la creación de camas y la petición de paquetes	29
C.4. Comunicación cliente servidor <i>websocket</i>	30
C.5. Diagrama de clases	31
C.6. Diagrama de componentes	32
C.7. Prototipo para escritorio	32
C.8. Prototipo para móvil	33
D.1. Árbol de directorios	36
D.2. Modelo de respuesta para la <i>API REST</i>	37
D.3. Formato del fichero de configuración <i>project.json</i>	42
D.4. Ejemplo de configuración de servicio en <i>Nginx</i>	42
D.5. Fichero de <i>systemd</i> para ejecutar el proyecto como un servicio	43
D.6. Código para ejecutar un <i>script Python</i> en un entorno	43
E.1. Manual: Ventana inicial	48
E.2. Manual: Ventana de <i>login</i>	48
E.3. Manual: Barra de navegación de administrador	49
E.4. Manual: Administración de las camas.	49
E.5. Manual: Crear cama	50
E.6. Manual: Modificar cama	51

E.7. Manual: Error al crear, modificar o borrar una cama.	52
E.8. Manual: Tabla de permisos por usuario.	52
E.9. Manual: Administración de los usuarios.	53
E.10. Manual: Formulario de creación de usuario.	54
E.11. Manual: Formulario de modificación de usuario.	54
E.12. Manual: Formulario de modificación de usuario.	55
E.13. Manual: Página con la información en tiempo real de una cama.	56
E.14. Manual: diferentes estados de un paciente	56
E.15. Manual: Barra de navegación, lado derecho.	56
E.16. Manual: Formulario de cambio de contraseña.	57

Índice de tablas

A.1. Costes de personal	10
A.2. Costes de <i>hardware</i>	11
A.3. Coste total	11
B.1. Caso de uso 1: Iniciar sesión	18
B.2. Caso de uso 2: Visualizar de datos	18
B.3. Caso de uso 2.1: Elegir cama	19
B.4. Caso de uso 2.2: Ver datos en tiempo real	20
B.5. Caso de uso 3: Administrar de usuarios	20
B.6. Caso de uso 3.1: Añadir usuarios	21
B.7. Caso de uso 3.2: Modificar contraseña	22
B.8. Caso de uso 3.3: Borrar usuario	22
B.9. Caso de uso 4: Administrar de camas	23
B.10. Caso de uso 4.1: Añadir cama	24
B.11. Caso de uso 4.2: Modificar cama	25
B.12. Caso de uso 4.3: Borrar cama	25
B.13. Caso de uso 4.4: Asignar cama a usuario	26
D.1. Especificaciones del API	40
D.2. Resultado de las pruebas sobre los transformadores	44
D.3. Resultados de los test unitarios sobre la <i>API</i>	45
D.4. Resultados de los tes sobre la interfaz	45

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En este apéndice se expondrán los distintos *sprints* que se han realizado y un estudio de la viabilidad del proyecto.

A.2. Planificación temporal

El proyecto se desarrolló siguiendo una metodología ágil basada en *Scrum*. Se dividió el progreso en *Sprints*, cada uno con una serie de tareas y su estimación en esfuerzo. Esta estimación o peso, se ha evaluado según la dificultad que se preveía tener, no como una medida horaria, esto se debe a que algunas tareas han sido sencillas para el desarrollador pero han requerido mayor tiempo de cómputo (e.g. Configurar experimentos), mientras otras tareas de ejecución mucho más rápida han requerido una mayor dedicación por el programador (e.g. Desarrollo de los test). Algunas tareas además se consideran épicas, estas son resúmenes de un proceso completo que abarca varias tareas y duran generalmente más de un *sprint*.

A continuación se mostrarán las listas de las tareas realizadas con los enlaces a las *issues* del repositorio [GitHub](#). Los *Sprints* fueron:

Sprint 1 - 10/11/18 hasta 20/11/18

En este primer *Sprint* solamente hubo dos tareas:

1. [Hacer exploración bibliográfica](#)

2. Configurar repositorio de Git

Sprint 2 - 21/11/18 hasta 05/12/18

Se desarrollaron cuatro tareas, todas de investigación, por lo que no hay *commits* asociados; las tareas fueron:

3. Lectura de «Automated Epileptic Seizure Detection Methods: A Review Study»
4. Exploración Bibliográfica - Orientado a caídas
5. Lectura del primer tema de "Minería de Datos"
6. Configurar VPN en Archlinux

Sprint 3 - 06/12/18 hasta 21/12/18

En este *sprint* se comenzó la documentación migrando las plantillas al repositorio y continuó la investigación a un área más técnica; las tareas fueron:

7. Iniciar documentación
8. Tabla de extracción de características
9. Búsqueda de librerías con funciones para el procesamiento de series temporales

Sprint 4 - 22/12/18 hasta 08/01/2019

Este *sprint* fue del aprendizaje de técnicas de minería de datos aplicada. Las tareas realizadas fueron:

10. Instalación de entorno Python
11. Graficar datos mediante PCA
12. Aprender el uso de librerías *Python* para la minería de datos

Sprint 5 - 09/01/2019 hasta 17/01/2019

En este *sprint* el objetivo fue probar distintas líneas de investigación para ver las características intrínsecas a los datos.

13. Configurar acceso al computador del grupo ADMIRABLE *Gamma*
14. Leer apuntes de Minería de Datos
15. Filtrado y suavizado de datos
16. Probar otras formas de proyección

Sprint 6 - 18/01/2019 hasta 24/01/2019

Tras descartar algunas proyecciones se exploraron las más prometedoras y se probaron nuevos filtros y detección de anomalías. Las tareas fueron por consiguiente:

17. Determinar el mejor preprocessamiento de los datos
18. Dibujado alrededor de las crisis
19. Probar formas de filtrado
20. Estudiar puntos clave de las proyecciones
21. Probar detección de anomalías por one-class

Sprint 7 - 25/01/2019 hasta 07/02/2019

Este *sprint* tuvo una duración mayor debido a que durante el periodo señalado se realizó un curso en la universidad donde se estudiaron las series temporales. Por este motivo, el servidor donde se han estado realizando las ejecuciones no estaba disponible retrasando la ejecución de los experimentos.

Las tareas se centraron en comprobar las proyecciones más interesantes con datos estadísticos además de profundizar en *One-Class* o detección de anomalías. Las tareas realizadas fueron:

22. Proyección LTSA con valores estadísticos [7]
23. Documentar Sprints del 1 al 6

24. Documentar investigación con Alicia Olivares
25. One Class - Mejorar la investigación de este apartado
26. Creación de transformadores de datos de *scikit-learn* para el procesamiento de los datos
27. Trasladar códigos existentes del procesado de datos a los transformadores de *scikit-learn* creados

Sprint 8 - 8/02/2019 hasta 14/2/2019

Este *sprint* fue dedicado a seguir desarrollando el estudio de *One Class* cuyos resultados se pueden ver en el apéndice *Cuaderno de Investigación*. Las tareas fueron:

28. Particionar los datos
29. Estudiar el funcionamiento de los algoritmos de *one-class*¹
30. Clasificador *One-Class* entrenando con datos de crisis
31. Clasificador *One-Class* entrenando con datos de no crisis
32. Preprocesado básico para *One Class*
 - a) Bruto
 - b) Filtrado *Butter* de 3 y 0.05
 - c) Filtrado *SavGol* de tamaño 15
 - d) Concatenación de estadísticos en ventana 25 sobre bruto
 - e) Concatenación de estadísticos con ventana 25 sobre *Butter* de 3 y 0.05
 - f) Concatenación de estadísticos con ventana 25 sobre *SavGol* de tamaño 15
33. Testear clasificadores con otras crisis

Las tareas de la 30 a la 33 se incluyeron en la tarea épica 29 aunque esta tuvo tareas del siguiente *sprint*

¹Tarea épica, siendo estas las tareas que engloban varias tareas más simples

Sprint 9 - 15/2/2019 a 21/2/2019

Este *sprint* trató de completar la linea de investigación de *One Class* abierta en la tarea épica 29 y documentar los resultados de *sprints* anteriores.

34. Documentar PCA y Proyecciones de 2-Variedad
35. Entrenamiento *One Class* con dos crisis
36. Testeo *One Class* con tercera crisis
37. Calcular área bajo la curva con entrenamiento de una clase
38. Visualización de constantes vitales

Son las tareas de la 35 a la 37 las que forman parte de la tarea épica 29.

Sprint 10 - 22/3/2019 a 28/3/2019

Este *sprint* estuvo centrado en la lectura y aprendizaje de nuevos métodos

39. Investigar sobre conjuntos de datos desequilibrados
40. Aprender Weka
41. Documentar *One Class*

Sprint 11 - 01/03/2019 a 21/03/2019

Este *sprint* tuvo una duración mayor debido a la diversidad de experimentos y la presencia de varios días no lectivos. Se centró en el lanzamiento de experimentos con *ensembles*.

42. Creación de test de transformers
43. Filtrado SMOTE
44. Pruebas con ensembles ya implementados
45. Prueba de ensembles para desequilibrados

Sprint 12 - 22/03/2019 - 28/03/2019

En esta ocasión comenzamos a reducir el esfuerzo sobre la investigación sobre un incremento en el diseño de la aplicación.

46. Exploración de herramientas
47. Documentar resultados anteriores
48. Diseño del servidor
49. Ejecutar experimentos Weka restantes

Sprint 13 - 29/03/2019 - 08/04/2019

Este *sprint* se centró en la definición de requisitos y de realizar experimentos con los resultados de Alicia Olivares. Sin embargo, estos experimentos no pudieron ser realizados al descubrir un fallo en la investigación previa, por lo que no se realizaron quedando pendientes ante nuevos datos. Por tanto, las tareas que realmente fueron realizadas fueron:

50. Crear la simulación de una cama para la obtención en tiempo real de datos
52. Diseño de casos de uso
53. Diseño de los datos para la API
54. Diseño de la base de datos
57. Diseño de la API
58. Requisitos funcionales

Sprint 14 - 09/04/2019 - 11/04/2019

Este *sprint* se concluyeron la creación de los experimentos que quedaron pendientes en el *sprint* anterior como seguir desarrollando componentes de la aplicación.

Sin embargo, muchas partes de este *sprint* fueron quedando en *icebox* tras algunas complicaciones en los experimentos y la nueva linea de investigación con *PRC* que se puede ver en el *Cuaderno de investigación* adjuntado.

51. Experimentos Weka con los datos de series temporales
55. Implementación de la base de datos
58. Creación de experimentos
59. Lanzamiento de experimentos
60. Documentación de resultados
61. Experimentos con conjuntos de datos desequilibrados²
62. Ultimar detalles del cuaderno de trabajo
63. Arreglar documentación de casos de uso
64. Completar diseño de bases de datos
65. Trabajar en formato de latex
66. Crear interfaz de la api
67. Creación y documentación de prototipos de interfaz

Sprint 15 - 12/04/2019 - 2/05/2019

En este *sprint* se comenzó la implementación de la API así como la realización de test automáticos de la misma. También se comenzó a preparar el servidor para que fuese accesible desde Internet.

68. Programación de la API para la gestión y distribución de datos de la aplicación
69. Test sobre la API
70. Acceso web a la API
71. Documentar Sprints anteriores
72. Configurar servidor
73. Entorno Flask

²Tarea épica

Sprint 16 - 3/05/2019 - 09/05/2019

Este *sprint* consistió en continuar el desarrollo del servidor.

74. Implementar hilo de procesamiento
75. Distribución de paquetes por SocketIO
76. Uso del clasificador de Alicia Olivares
77. Actualización de Chart.js es muy lenta [BUG]
78. Lanzar proyecto en el servidor

Sprint 17 - 10/05/2019 - 16/05/2019

Este *sprint* se centró en el desarrollo de la vista así como reparar algunos *bugs* que fueron apareciendo. También se intentó utilizar *Heroku* pero al conseguir eliminar las limitaciones del servidor se desechó la idea.

80. Crear conexiones SQL bajo demanda por `request`
81. Hacer todas las ventanas
82. Crear cama simulada de los datos alrededor de las crisis
83. Escribir memoria - Herramientas
84. Error en conexiones por request: MySQL Connection Not Available [BUG]

Sprint 18 - 17/05/2019 - 30/05/2019

Este *sprint*, que duró dos semanas, se centró en hacer algunas mejoras sobre la aplicación así como documentar algunas partes de la memoria.

85. Guardar probabilidades del imblearn
86. Censurar opciones de administración de camas
87. Sistema de alertas y control en frontend
88. Simular todas las camas con los mismos hilos
89. Crear un clasificador aleatorio

90. Escribir el abstract de la memoria
91. Documentar proceso de instalación
92. Concluir ventanas restantes

Spint 19 - 31/05/2019 - 13/06/2019

Este *sprint*, que duró dos semanas, debido a los exámenes finales que ocurrieron en este periodo, se centró en continuar la documentación así como mejorar la interfaz de la aplicación web.

93. Continuar la documentación
94. Reparar varios bugs
95. Diseñar los test de selenium
96. Crear pantalla de carga
97. Migrar la sesión y la gestión de usuarios a flask-session y flask-login

Spint 20 - 14/06/2019 - 20/06/2019

Este penúltimo *sprint* se centró en terminar todos los puntos de la documentación.

98. Corregir anexo y memoria
99. Escribir objetivos
100. Escribir conceptos teóricos
101. Escribir estudio de viabilidad
102. Escribir manual del usuario
103. Escribir trabajos relacionados
104. Escribir conclusiones y líneas futuras
105. Escribir aspectos relevantes

Concepto	Coste(€)
Salario mensual neto	1.225,7 [3]
Retención IRPF (15 %)	216,3
Seguridad Social (28,3 %)	569,16
Salario mensual bruto	2.011,16
Total 7 meses y dos empleados	28.156,24

Tabla A.1: Costes de personal.

Spint 21 - 21/06/2019 - 27/06/2019

En este último *sprint* se realizaron las últimas partes del proyecto como las correcciones de algunos bugs y de la documentación

- 106. Documentar resultados de las pruebas de usabilidad
- 107. Arreglar últimos detalles de la memoria y anexos
- 108. Reparar *bug* concurrencia
- 109. Añadir el cuaderno de investigación

A.3. Estudio de viabilidad

Viabilidad económica

Debido a que este TFG se ha realizado en un proyecto de investigación de varias personas, este apartado será común al del TFG de Alicia Olivares Gil, al ser parte del equipo de investigación.

Costes

Los costes de personal se desglosarán en las siguientes categorías: costes de personal para el salario de dos personas, el coste hardware de las herramientas utilizadas, coste software del software utilizada. También se incluirán gastos aproximados del material que se ha dispuesto que ya se poseía.

El coste de personal se puede ver en la tabla A.1

Para el desarrollo de este proyecto no se ha adquirido ningún *hardware* nuevo, por lo que únicamente se incluirán los costes del material con el que ya se contaba asumiendo una amortización en 5 años, y calculando solo el

Concepto	Coste(€)	Coste amortizado(€)
Dispositivo móvil	150	17,5
Ordenador portátil (x2)	800	93,33
<i>MainFrame</i>	3.000	350
GPU (x3)	4.500	525
Total	8.450	985,83

Tabla A.2: Costes de *hardware*.

Concepto	Coste(€)
Coste de personal	28.156,24
Coste del <i>hardware</i>	985,83
Total	29.142,07

Tabla A.3: Coste total.

coste de amortización correspondiente a la duración del proyecto (7 meses). Se puede ver en la tabla A.2

No ha habido costes software ya que todas las herramientas utilizadas han sido gratuitas o de código abierto.

Teniendo en cuenta los costes de personal y de *hardware*, el coste económico total se puede ver en la tabla A.3

Viabilidad legal

En este proyecto se ha realizado con la ayuda de software de terceros con licencias propias que influyen sobre la viabilidad legal del proyecto.

El software utilizado se puede encontrar en el apartado ‘Técnicas y herramientas’ de la memoria. Las licencias que utilizan son:

- **MIT:** Esta licencia permite el uso comercial del producto, la modificación del mismo, la libre distribución y el uso privado. No tiene garantías ni responsabilidad. La única condición es hacer referencia a ella. Como no obliga a mantener la licencia ni afecta a la distribución del software que use la licencia final del producto, esta puede ser cualquiera.

- **AGPL:** Esta licencia permite el uso comercial, la libre distribución y modificación. Así también permite el uso de patentes y el uso privado. Entre sus condiciones están que el software debe distribuirse con su código, hacer referencia a la licencia, se deben documentar los cambios en el código y utilizar la misma licencia o una compatible (como GPLv3). Tiene en sus limitaciones la garantía y la responsabilidad.
- **GPLv2:** Tiene las mismas características que AGPL, cambia la cláusula de patentes sin prohibirlas, no especifica nada de ellas. Tampoco obliga a distribuir el código vía aplicación web.
- **BSD Simplified:** Tiene características semejantes a MIT en el contexto en el que la usamos.
- **BSD 3-Clause:** Tiene características semejantes a MIT en el contexto en el que la usamos.

Entre todas estas licencias la más restrictiva es AGPL que será la que se use en esta aplicación y todas son compatibles entre si.

Apéndice B

Especificación de Requisitos

B.1. Introducción

El primer paso de todo desarrollo de software es definir bien los requisitos de la aplicación así como los detalles generales que se necesitan de la aplicación. En este apartado se desarrollaran los requisitos y los casos de uso.

B.2. Objetivos generales

La aplicación web tiene lo siguientes objetivos:

- Monitorizar en tiempo real a los pacientes con epilepsia.
- Gestionar los distintos accesos del personal médico a los datos de los pacientes.
- Ofrecer una API REST de la aplicación.
- Crear una interfaz web fácil e intuitiva.

B.3. Catálogo de requisitos

En esta sección se presentan los requisitos funcionales y los no funcionales. Tanto los requisitos funcionales como su especificación de los mismos en la sección [B.4](#) son comunes a los de Alicia Olivares Gil debido a que la aplicación desarrollada está adscrita al mismo proyecto de investigación y la base de la aplicación a realizar es la misma.

Requisitos funcionales

- **RF-1 Confidencialidad del sistema:** solamente usuarios autorizados podrán acceder al sistema.
 - **RF-1.1 Identificación de usuario:** los usuarios se identificarán con un *nickname* y una contraseña
 - **RF-1.2 Rol de administración:** existirá un usuario especial que podrá administrar el sistema completamente sin restricciones.
 - **RF-1.3 Visualización de una cama:** los usuarios validados deben poder observar los datos en tiempo real de las camas disponibles.
 - **RF-1.4 Restricción de acceso:** los usuarios solamente podrán tener acceso a los datos de las camas permitidas.
 - **RF-1.5 Acceso completo al administrador:** el administrador debe poder acceder a todas las camas existentes.
- **RF-2 Gestión de las camas:** el administrador ha de gestionar las camas pudiendo añadir, modificar y borrar.
 - **RF-2.1 Añadir cama:** el administrador ha de poder añadir una nueva cama al sistema.
 - **RF-2.2 Modificar cama:** el administrador ha de poder modificar los datos una cama existente.
 - **RF-2.3 Borrar cama:** el administrador ha de poder borrar una cama del sistema.
 - **RF-2.4 Asignar camas a usuarios:** el administrador se encarga de decidir que usuario puede acceder a que cama.
- **RF-3 Gestión de los usuarios:** el administrador ha de gestionar los usuarios pudiendo añadir, modificar y borrar.
 - **RF-3.1 Añadir usuario:** el administrador ha de poder añadir un nuevo usuario al sistema.
 - **RF-3.2 Modificar usuario:** el administrador ha de poder modificar los datos un usuario existente.
 - **RF-3.3 Borrar usuario:** el administrador ha de poder borrar un usuario del sistema.
- **RF-4 Visualización de los datos:** los usuarios han de poder ver de las camas disponibles el estado actual del paciente, sus constantes vitales y las presiones.

Requisitos no funcionales

- **RNF-1 Usabilidad:** la aplicación debe cumplir estándares de usabilidad teniendo una curva de aprendizaje baja y un uso de metáforas adecuado.
- **RNF-2 Disponibilidad:** las camas existentes han de ser siempre accesibles por sus usuarios asociados y dar una información correcta de su estado
- **RNF-3 Confidencialidad:** los datos de las camas, al ser en parte constantes vitales de pacientes, solamente han de ser accesibles por los usuarios autorizados.
- **RNF-4 Escalabilidad:** el sistema debe ser escalable para adaptarse mejor a un incremento de carga del sistema.
- **RNF-5 Seguridad:** los usuarios deben poder identificarse sólidamente con el sistema sin que sus datos o sus credenciales (*tokens*) sean accesibles por terceros, incluso el administrador.
- **RNF-6 Extensibilidad:** la API del sistema debe ser fácilmente extensible a nuevas funcionalidades incorporando de manera eficaz soporte a nuevas peticiones.
- **RNF-7 Persistencia:** los servicios de procesamiento de las camas activas deben mantenerse funcionando aunque no existan clientes activos para evitar retrasos muy altos ante nuevas conexiones.
- **RNF-8 Fiabilidad:** los datos de la aplicación son correctos y actuales además de garantizar una predicción óptima del estado del paciente.

B.4. Especificación de requisitos

Los requisitos funcionales generan un conjunto de casos de uso que serán la base del desarrollo de la aplicación. La especificación de los mismos se encuentran entre la tabla B.1 y la tabla B.12. La representación gráfica se puede ver en los diagramas de las figuras B.1, B.2, B.3 y B.4. Existen dos actores, el **administrador** que se encarga de toda la labor de gestión tanto de usuarios como de camas y el **usuario** que únicamente puede gestionarse a sí mismo y ver los datos de las camas que tenga permitidas.

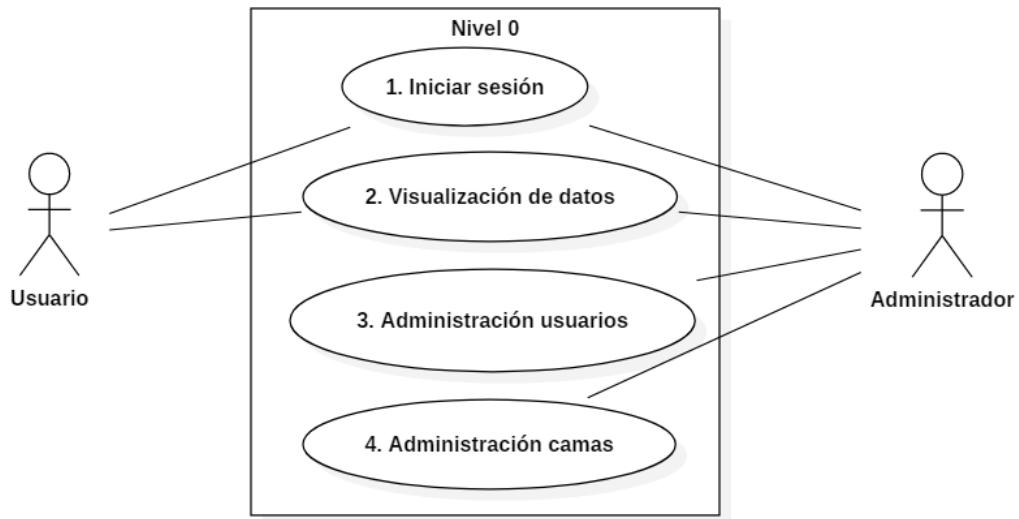


Figura B.1: Diagrama casos de uso - Nivel 0

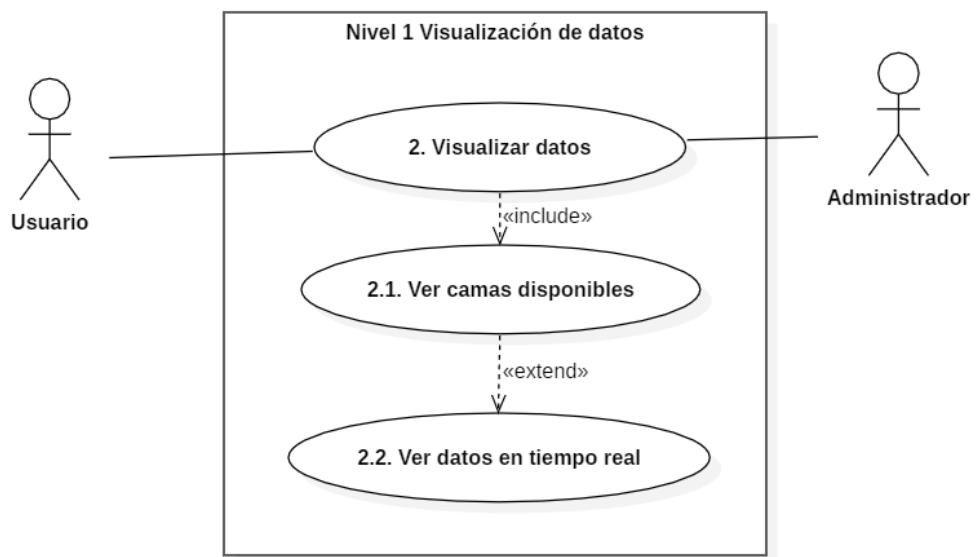


Figura B.2: Diagrama CU-2 - Nivel 1

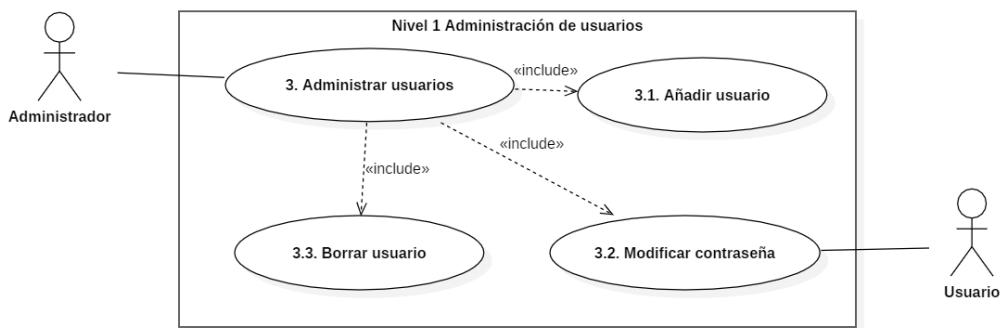


Figura B.3: Diagrama CU-3 - Nivel 1

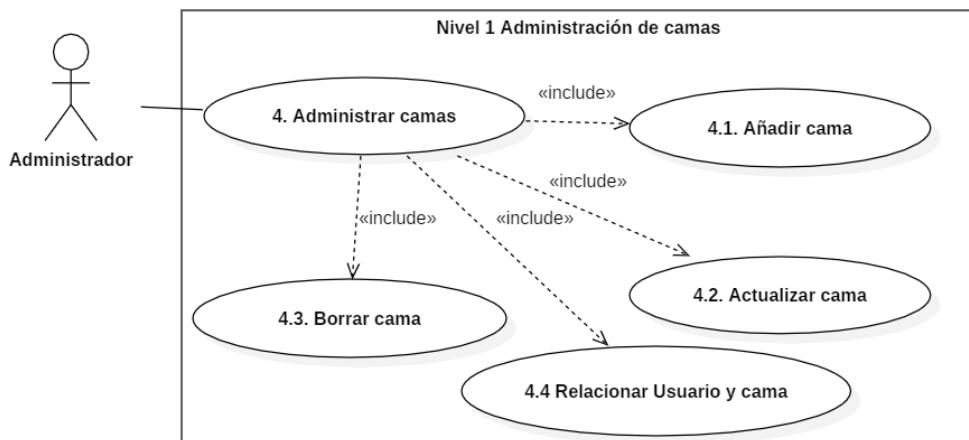


Figura B.4: Diagrama CU-4 - Nivel 1

CU-1: Iniciar sesión		
Descripción	El usuario se identifica en el sistema	
Precondiciones	No existe una sesión activa válida	
Requisitos	RF-1, RF-1.1	
Usuario	Anónimo	
	Paso	Acción
Secuencia normal	1	El cliente envía sus credenciales al servidor
	2	El servidor acepta las credenciales devolviendo el token de sesión
Postcondiciones	El usuario tiene una sesión activa válida	
Excepciones	Paso	Acción
	2	Si las credenciales son incorrectas el servidor responde con error
Frecuencia	Alta	
Importancia	Crítico	
Comentarios	Es siempre lo primero que aparecerá	

Tabla B.1: Caso de uso 1: Iniciar sesión

CU-2: Visualizar de datos		
Descripción	Ver lista de las camas disponibles	
Precondiciones	Sesión activa válida	
Requisitos	RF-1.3, RF-1.4	
Usuario	Administrador y Usuario	
	Paso	Acción
Secuencia normal	1	El cliente solicita ver las camas disponibles
Postcondiciones	El cliente está en la pantalla de camas disponibles	
Frecuencia	Alta	
Importancia	Alta	

Tabla B.2: Caso de uso 2: Visualizar de datos

CU-2.1: Elegir cama

Descripción	Elegir cama	
Precondiciones	Sesión activa válida	
Requisitos	RF-1.3, RF-1.4, RF-4	
Usuario	Logueado	
	Paso	Acción
Secuencia normal	1	El cliente solicita ver las camas disponibles
	2	El servidor abre conexiones paralelas para actualizar en tiempo real el estado de las camas
	3	El cliente decide que cama ver
Postcondiciones	El cliente entra en la ventana de los datos en tiempo real	
Frecuencia	Alta	
Importancia	Alta	

Tabla B.3: Caso de uso 2.1: Elegir cama

CU-2.2: Ver datos en tiempo real		
Descripción	Ver datos en tiempo real	
Precondiciones	Sesión activa válida y cama existente y accesible	
Requisitos	RF-1.3, RF-1.4, RF-4	
Usuario	Administrador y usuario	
	Paso	Acción
Secuencia normal	1	El cliente solicita una nueva conexión
	2	El servidor provee una conexión en tiempo real con los datos
Postcondiciones	El usuario tiene una conexión paralela abierta con los datos en tiempo real	
Excepciones	Paso	Acción
	2	Si un paquete faltase o la señal fuera, débil se alertaría al usuario
Frecuencia	Alta	
Importancia	Máxima	

Tabla B.4: Caso de uso 2.2: Ver datos en tiempo real

CU-3: Administrar de usuarios		
Descripción	Administración de usuario: alta, baja y modificación	
Precondiciones	Sesión de administrador válida	
Requisitos	RF-3	
Usuario	Administrador	
	Paso	Acción
Secuencia normal	1	El administrador entra en el menú de administración de usuarios
Postcondiciones	El administrador está en el menú de administración de usuarios	
Frecuencia	Baja	
Importancia	Alta	

Tabla B.5: Caso de uso 3: Administrar de usuarios

CU-3.1: Añadir usuarios		
Descripción	Añadir usuarios	
Precondiciones	Sesión de administración activa	
Requisitos	RF-3.1	
Usuario	Administrador	
	Paso	Acción
Secuencia normal	1	El administrador elige añadir un nuevo usuario
	2	Se introduce un nombre de usuario para identificarlo
	3	Se introduce una contraseña dos veces
	4	Se almacenan los datos
Postcondiciones	Existe un nuevo usuario en el sistema	
Excepciones	Paso	Acción
	2	Si el nickname existiese
	3	La contraseña añadida no coincide en las dos ocasiones
Frecuencia	Baja	
Importancia	Alta	

Tabla B.6: Caso de uso 3.1: Añadir usuarios

CU-3.2: Modificar contraseña		
Descripción	Cambiar la contraseña de un usuario	
Precondiciones	Sesión activa válida, usuario existente	
Requisitos	RF-3.2	
Usuario	Administrador y Usuario	
	Paso	Acción
Secuencia normal	1	Si es usuario normal ir a 3
	2	Si es administrador elegir a qué usuario cambiar la contraseña
	3	Se introduce una contraseña nueva dos veces
	4	Se actualizan los datos
Postcondiciones	La contraseña ha cambiado	
Excepciones	Paso	Acción
	3	La contraseña añadida no coincide en las dos ocasiones
Frecuencia	Baja	
Importancia	Alta	

Tabla B.7: Caso de uso 3.2: Modificar contraseña

CU-3.3: Borrar usuario		
Descripción	Elimina un usuario de la base de datos	
Precondiciones	Sesión de administración válida, usuario existente	
Requisitos	RF-3.3	
Usuario	Administrador	
	Paso	Acción
Secuencia normal	1	Elegir a que usuario (no administrador) eliminar
	2	Eliminar usuario y todos los datos vinculados
Postcondiciones	El usuario ha sido eliminado	
Frecuencia	Baja	
Importancia	Media	

Tabla B.8: Caso de uso 3.3: Borrar usuario

CU-4: Administrar de camas

Descripción	Administración de camas: alta, baja, modificación y asignación a usuarios	
Precondiciones	Sesión de administración válida	
Requisitos	RF-2	
Usuario	Administrador	
	Paso	Acción
Secuencia normal	1	El administrador entra en el menú de administración de camas
Postcondiciones	El administrador está en el menú de administración de camas	
Frecuencia	Baja	
Importancia	Media	

Tabla B.9: Caso de uso 4: Administrar de camas

CU-4.1: Añadir cama		
Descripción	Añadir cama	
Precondiciones	Sesión de administración válida	
Requisitos	RF-2.1	
Usuario	Administrador	
	Paso	Acción
Secuencia normal	1	El administrador elige añadir una nueva cama
	2	Se introduce el grupo multicast de la cama (IP y Puerto)
	3	Se introduce el nombre identificador
	4	Se almacenan los datos
Postcondiciones	Existe una nueva cama en el sistema	
Excepciones	Paso	Acción
	2	El grupo multicast pertenece a otra cama
	3	El nombre identificativo existe para otra cama
Frecuencia	Media	
Importancia	Crítica	
Comentarios	El grupo multicast se configura en la cama y el administrador solamente debe conocerlo, no configurar la cama física	

Tabla B.10: Caso de uso 4.1: Añadir cama

CU-2.2: Modificar cama		
Descripción	Modificar los datos de la cama	
Precondiciones	Sesión de administración válida, cama existente	
Requisitos	RF-2.2	
Usuario	Administrador	
	Paso	Acción
Secuencia normal	1	Se elige que cama modificar
	2	Se actualizan los datos a conveniencia del administrador según CU-4.1
	4	Se actualizan los datos
Postcondiciones	Los datos de la cama se modifican	
Excepciones	Paso	Acción
	2	Mismas excepciones que en CU-4.1
Frecuencia	Baja	
Importancia	Alta	

Tabla B.11: Caso de uso 4.2: Modificar cama

CU-4.3: Borrar cama		
Descripción	Elimina una cama de la base de datos	
Precondiciones	Sesión de administrador válida, cama existente	
Requisitos	RF-2.3	
Usuario	Administrador	
	Paso	Acción
Secuencia normal	1	Elegir a que cama eliminar
	2	Eliminar cama y todos los datos vinculados
Postcondiciones	La cama ya no está en la base de datos	
Frecuencia	Baja	
Importancia	Media	

Tabla B.12: Caso de uso 4.3: Borrar cama

CU-4.4: Asignar cama a usuario

Descripción	Permite a un usuario ver los datos de una cama o quitar ese permiso	
Precondiciones	Sesión de administración válida, cama y usuario existentes	
Requisitos	RF-2.4	
Usuario	Administrador	
	Paso	Acción
Secuencia normal	1	Elegir cama
	2	Elegir usuario
	3	Si la relación existe se puede eliminar el permiso
	3	Si la relación no existe se puede crear el permiso
Postcondiciones	El usuario tiene acceso a la cama, o pierde el mismo	
Frecuencia	Media	
Importancia	Crítica	

Tabla B.13: Caso de uso 4.4: Asignar cama a usuario

Apéndice C

Especificación de diseño

C.1. Introducción

Tras el estudio de los requisitos se pasa a un diseño, en este apéndice se explicarán los distintos diseños de la aplicación, desde los datos, los procesos más importantes, el diseño arquitectónico y las interfaces.

C.2. Diseño de datos

De los requisitos y casos de usos se deduce el diseño de los datos. Para poder cumplir correctamente con las necesidades del cliente se introducen dos entidades, los **usuario** y las **camas** en la que cada uno almacena la información relevante propia. A su vez se relacionan en el sistema de permisos de qué persona puede ver qué cama. El diagrama Entidad-Relación se puede observar en la figura C.1.

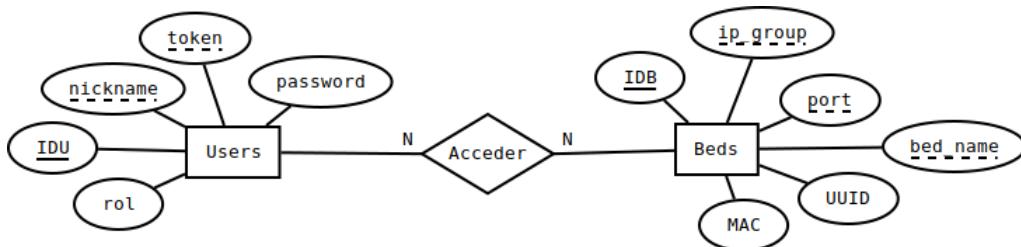


Figura C.1: Diagrama entidad-relación

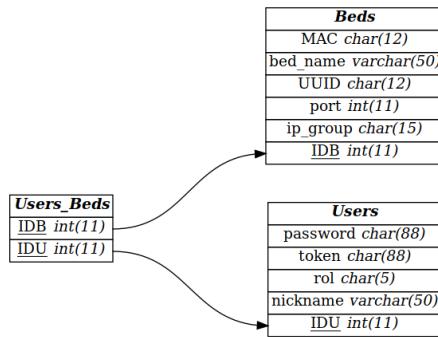


Figura C.2: Diagrama relacional

De este diagrama se genera el diagrama relacional (figura C.2) en el que se especifican las tablas que se usarán en el producto final.

C.3. Diseño procedimental

Las clases principales para entender el funcionamiento explicado más adelante son:

- **BedListener**: esta clase se encarga de escuchar un grupo *multicast* (par IP del tipo D y puerto) y encolar los paquetes recibidos.
- **BedProcess**: esta clase se responsabiliza de procesar los datos encolados por el **BedListener**, preprocesarlos y realizar las predicciones. Con estos resultados genera un paquete y lo encola.
- **BedBroadcaster**: esta clase tiene la responsabilidad de distribuir los paquetes encolados por el **BedProcess** a los clientes que lo soliciten.

Existen diversos procesos importantes en la aplicación. El primero es la creación de los componentes que se encargan de monitorizar las camas, como se puede observar en la figura C.3. Al principio, el servidor instancia todos los **BedListener** necesarios según las camas que existan, y estos instancian su **BedProcess** correspondiente. Es importante destacar que estas dos clases funcionan como hilos.

Por otra parte el cliente abre conexiones con el servidor solicitando datos, este genera un hilo del **BedBroadcaster** que existe mientras el clien-

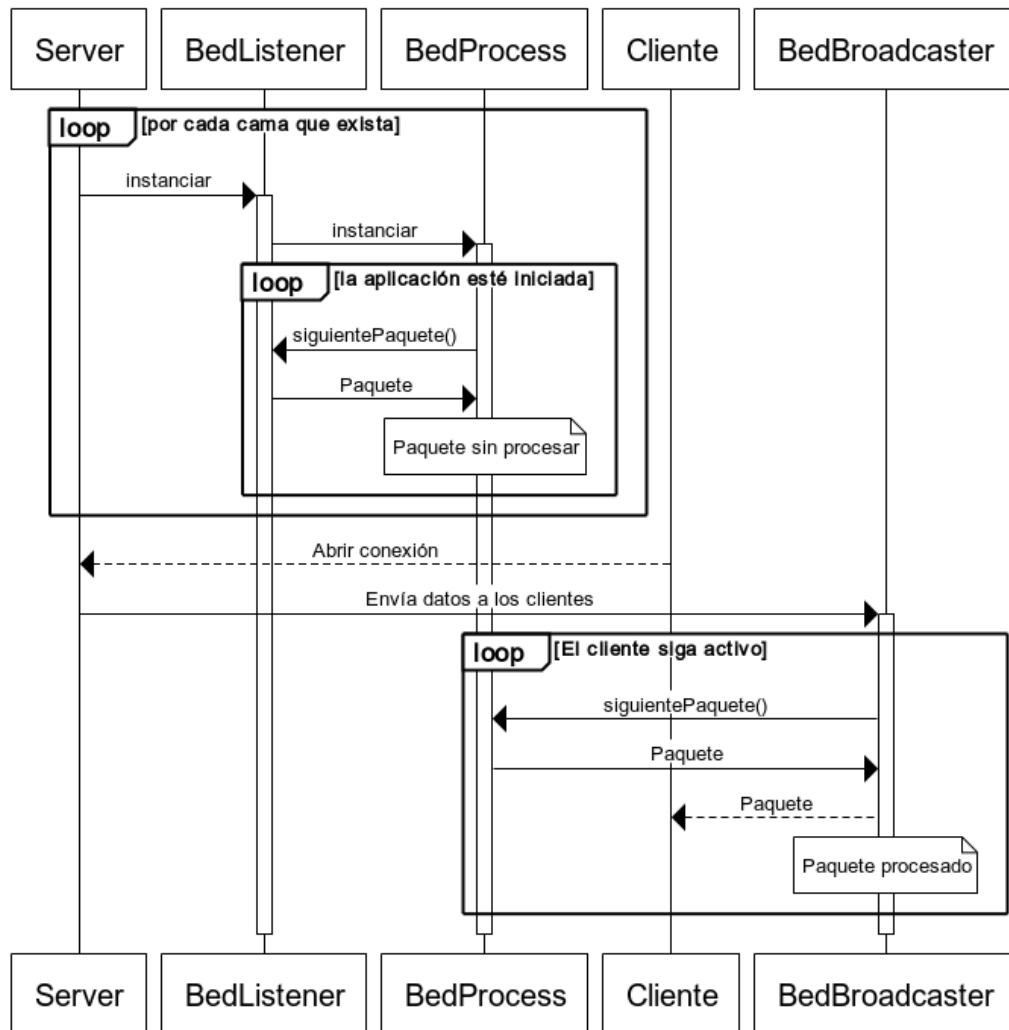


Figura C.3: Diagrama de secuencia de la creación de camas y la petición de paquetes

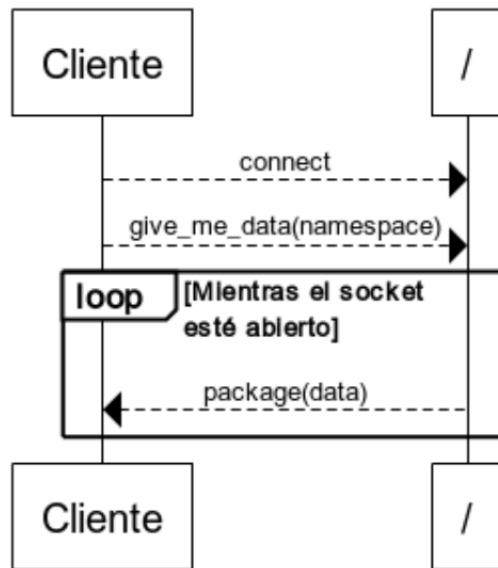


Figura C.4: Comunicación cliente servidor *websocket*

te esté activo (el que comenzó la petición o cualquiera que se incorpore posteriormente).

El proceso completo de la conexión del cliente se puede ver en la figura C.4. En esta cada llamada es un **evento** de *socketio* y los argumentos los datos que espera. El proceso de esta conexión es:

1. El **cliente** realiza una conexión a *socketIO*.
2. Cuando se capture el evento **connect** se emite un mensaje al servidor con el evento **give_me_data** y el **namespace** de la cama solicitada.
3. Por último se debe dejar un *listener* sobre el evento **package** en el **namespace**. Este evento es emitido por el servidor y contiene los datos que se pueden ver en la tabla D.1 para el caso de uso 2.2 para *WebSocket*.

Los procedimiento de toda la *API* se especifican más adelante en la sección D.3.

C.4. Diseño arquitectónico

La arquitectura software sigue el patrón *MVC* [4]. Sin embargo, al ser un problema sencillo no se traslada a un diseño literal de este patrón. El

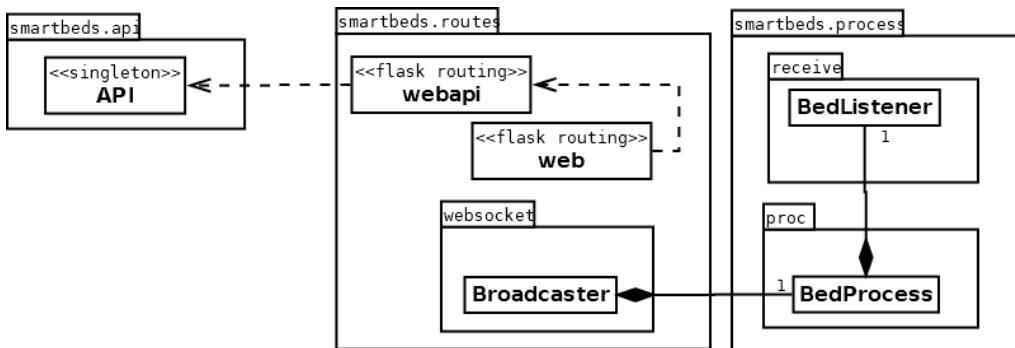


Figura C.5: Diagrama de clases

modelo se mantiene en la base de datos y no tiene una representación en ninguna clase. El *controlador* está pensado en varios componentes, una *API* que gestiona todos los accesos a la base de datos y un *proxy* que recoge las peticiones web y las preprocesa para redirigirlas a la *API*. Por último, la *vista* puede ser completamente diseñada para la situación que se requiera, en esta aplicación se gestiona mediante plantillas *HTML* y peticiones *AJAX* ya que se trata de una aplicación web. Gracias a que el controlador funciona como una *API REST* esta vista puede ser cambiada según se deseé.

Otra arquitectura relevante es el *pipeline* de procesamiento de los datos de pacientes. Se puede observar, junto con los módulo de *routing*, en la figura C.5. Se trata de tres componentes, dos con contexto en toda la aplicación, que se trata del *listener* de la cama y el procesador de la misma. El último es el *broadcaster* que emite los datos a los pacientes. En la sección C.3 se puede ver la comunicación del cliente y el proceso de inicialización de los componentes de este *pipeline*.

La arquitectura global de la aplicación sigue el diagrama de componentes de la figura C.6. La aplicación, desarrollada en la librería de *Python*, *Flask*, se conecta a una base de datos relacional compatible con la API de *MySQL*, en el caso concreto del despliegue actual se usa la implementación de *MariaDB*; la aplicación también lanza hilos para la escucha de camas mediante multidifusión, aunque el diagrama solo tiene una pueden ser tantas como se requiera. El último módulo es el servidor web, en el caso del despliegue actual es un *Nginx* pero puede ser cualquiera, lo que debe ser de proxy reverso.

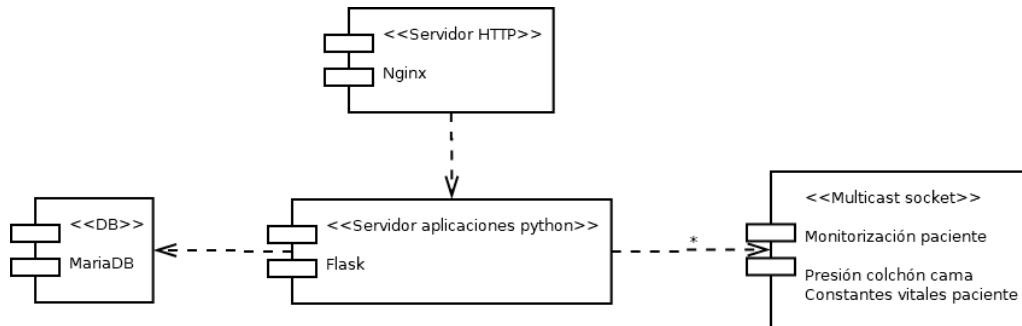


Figura C.6: Diagrama de componentes

C.5. Diseño de interfaces

Las interfaces se han diseñado teniendo en cuenta la usabilidad de la misma, así como un diseño simple que permitiese que fuese más intuitiva con una curva de aprendizaje leve. Se ha tenido en cuenta también que la interfaz sea adaptable a una gran variedad de pantallas teniendo en cuenta que, al ser una aplicación web, el uso de la misma puede ser en multitud de dispositivos diferentes (diseño *responsive* [5]).

Los primeros prototipos fueron realizados sobre el caso de uso especificado en la tabla B.4, tanto para escritorio (imagen C.7) como para móvil (imagen C.8).

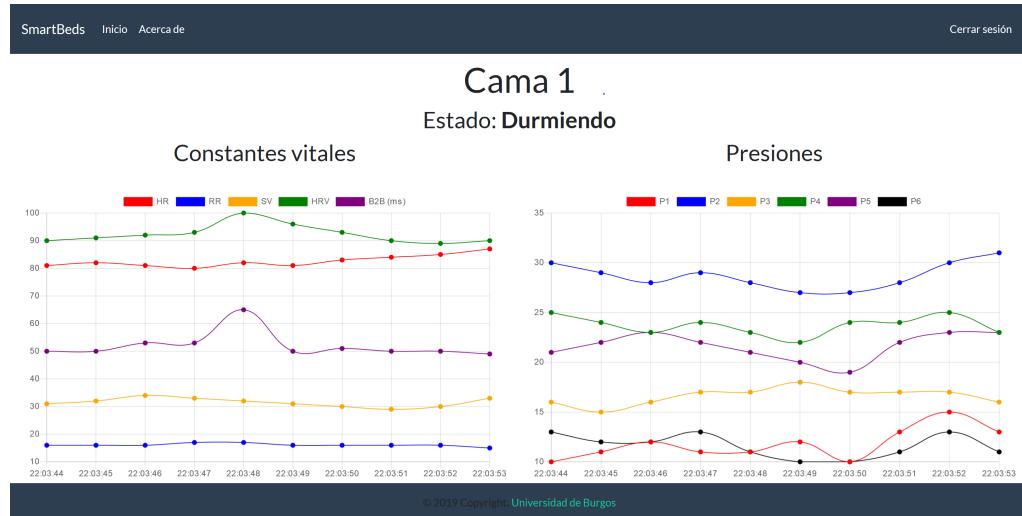


Figura C.7: Prototipo para escritorio

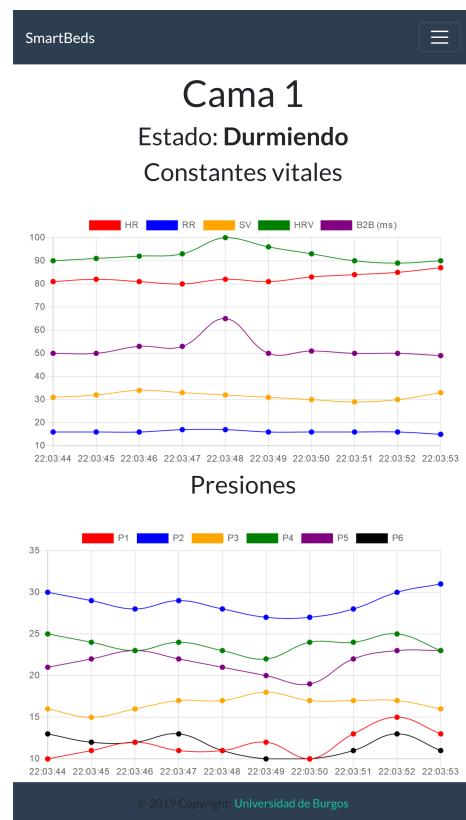


Figura C.8: Prototipo para móvil

Apéndice D

Documentación técnica de programación

D.1. Introducción

En este apéndice se explicarán todos los detalles necesarios para poder entender los componentes software que existen así como como desarrollar aplicaciones siguiendo la *API REST* provista.

D.2. Estructura de directorios

En la carpeta `doc` se encuentra toda la documentación de este proyecto. En `research` están los ficheros de la investigación, dentro de esta están en el directorio `data`¹ que contiene los datos proporcionados para realizar el estudio, `model` con los modelos binarios, `src` con los ficheros fuentes y los *Jupyter Notebooks* con la investigación y en `tests` con los tests sobre algunas fuentes.

La carpeta `smartbes` están las fuentes de la aplicación web siendo el modulo raíz. Dentro de esta están los módulo `process` con los algoritmos de procesamiento de los datos, `resources` con la configuración y los `assets` de la página web (plantillas, códigos *JavaScript* etc), `routes` que engloba las rutas *URI* para dirigir las peticiones. Por último, está la carpeta `test` con las pruebas unitarias.

¹Esta carpeta no se encuentra en el repositorio *GitHub* porque contienen datos privados

APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN

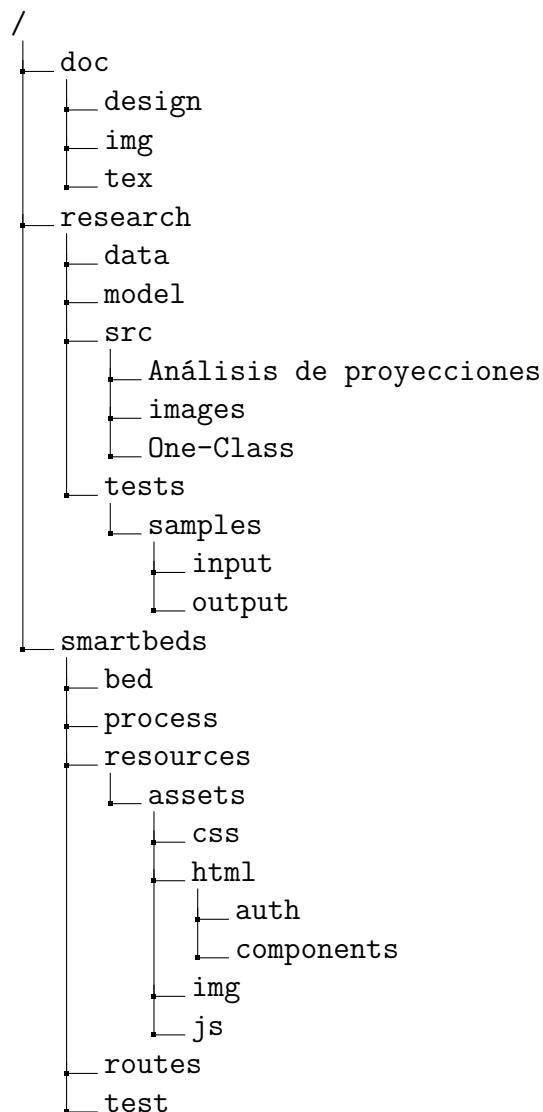


Figura D.1: Árbol de directorios

De manera especial se encuentra la carpeta `bed` dentro de `smartbeds` que contiene un *script* para la simulación de camas.

El árbol de directorios del proyecto se puede ver en la figura D.1

```

1 {
2 "status": 200|400|401|403|404|418|500,
3 "message": "OK"|"Error description"
4 }
```

Figura D.2: Modelo de respuesta para la *API REST*

D.3. Manual del programador

API

Para proveer los servicios de esta aplicación a nuevos entornos se incorpora una API para utilizar los diferentes servicios del sistema especificados en los Casos de Uso, sección B.4.

El funcionamiento general de la API serán peticiones POST mediante application/x-www-form-urlencoded ante rutas específicas con los datos requeridos para cada petición. Y según sea el caso el servidor contestará con un fichero JSON con la respuesta adecuada. De manera particular está el sistema en tiempo real que funciona mediante mensajes de *WebSockets* [6] usando la librería *SocketIO* [1] mediante la serie de eventos que se pueden ver en la figura C.4 de la sección C.3 .

Todas las respuestas del servidor contendrán los campos que se puede ver en el código de la figura D.2.

El valor de **status** tendrá un valor según los códigos HTTP definidos en el RFC 7231 [2] y el mensaje será una explicación detallada del error producido.

Las distintas peticiones se especifican en la tabla D.1.

CU	URI	Petición	Respuesta
CU-1	/api/auth	"user": text, "pass": text	{ ... "token": text, "role": text, "username": text }

continúa en la página siguiente

APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN

continúa desde la página anterior

CU	URI	Petición	Respuesta
CU-2.1	/api/beds	"token": text	<pre> "token": "...", "beds": [{ "bed_name": text, "ip_group": text, "port": text, "MAC": text, "UUID": text }] </pre>
CU-2.2	/api/bed	token=text& bedname= text	<pre> token=text& bedname= text { "namespace": text } </pre>
CU-2.2	/ (WebSocket)	{ "	<pre> { "results": [result, namespace prob, ": press_state, namespace hr_state], "instance": datetime "vital": [HR, RR, SV, HRV, B2B], "pressure": [P1, P2, .., P6]] } </pre>

continúa en la página siguiente

continúa desde la página anterior

CU	URI	Petición	Respuesta
CU-3	/api/users	token=text	{ ... "users": [text, ..., text] }
CU-3.1	/api/user/add	token=text & username= text & password= text & password-re =text	{ ... }
CU-3.2	/api/user/mod	token=text & username= text & password= text & password-re =text [&password- old=text]	{ ... }
CU-3.3	/api/user/del	token=text & username= text	{ ... }
CU-4.1	/api/bed/add	token=text & bed_name= text & ip_group= text & port=text & MAC=text & UUID=text	{ ... }

continúa en la página siguiente

APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN

continúa desde la página anterior

CU	URI	Petición	Respuesta
CU-4.2	/api/bed/mod	token=text& bed_name= text& ip_group= text& port=text& MAC=text& UUID=text	{ ... }
CU-4.3	/api/bed/del	token=text& bed_name= text	{ ... }
CU-4.4	/api/bed/perm	token=text& mode=[info change] [&bed_name= text& username= text]	{"permission": [{ "username": text, "bed_name": text }, ...] }

Tabla D.1: Especificaciones del API

D.4. Compilación, instalación y ejecución del proyecto

Este proyecto contiene un servidor web, del cual se explicará su proceso de despliegue en este apartado. Sin embargo, no disponemos de camas reales que puedan aportar datos en tiempo real como hemos visto en la figura C.6, debido a que este *hardware* es privado.

En su lugar se ha creado un *script* de *Python* que emite cada 0,4 segundos de manera indefinida datos que se han obtenido previamente. Este *script* se encuentra en la carpeta **smartbeds/bed**.

D.4. COMPILACIÓN, INSTALACIÓN Y EJECUCIÓN DEL PROYECTO

Requisitos del sistema

Para el funcionamiento correcto de la aplicación se requiere de *hardware* los siguientes mínimos:

- **Procesador:** arquitectura de 64 bits con soporte multihilo. Se recomienda un mínimo de 1,5 GHz, 16MiB de memoria caché y dos núcleos.
- **Memoria:** 768MiB, incorporando 256MiB aproximadamente por cada nueva cama incorporada.
- **Almacenamiento:** 16MiB aproximadamente.

Requisitos software

Para poder ejecutar la aplicación se necesitan los siguientes apartados de software.

Entorno *Python*, versión 3.7 con las librerías de *requeriments.txt* instaladas (`pip install -r requeriments.txt`). Base de datos *MySQL* o con API compatible como *MariaDB*. Servidor web con soporte de *proxy* reverso y *websokets* como *Nginx* versión 1.16.

Instalación en entorno GNU/Linux

El primer paso necesario es la configuración de la base de datos en la cual se van a almacenar los datos del sistema. La estructura de esta base de datos se encuentra en el fichero `/smartbeds/resources/database.sql`. Tras esto es necesario crear el fichero `/smartbeds/resources/project.json` que tendrá la configuración de la aplicación siguiendo el formato de la figura D.3

En cada campo habría que incorporar los cambios oportunos y se recomienda no modificar el campo `url`. De manera especial el campo `mode` puede ser `ssl` o `no-ssl` si los *websockets* se ejecutan sobre *SSL* o no. Tras esto ya el programa es ejecutable y puede funcionar al ejecutar el fichero `/index.py`.

El siguiente paso es crear la configuración del servidor *HTTP* para redirigir las peticiones externas a la aplicación local. Como ejemplo, para un servidor *Nginx* la configuración se puede ver en la figura D.4

Siendo necesario modificar el nombre del servidor por el dominio o IP de acceso y la raíz como la ruta donde se encuentra el fichero `index.py`, aunque

APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN

```
1 {
2     "secret-key": "clave secreta",
3     "url": "http://127.0.0.1",
4     "port": 3031,
5     "mode": "ssl"
6     "database": {
7         "host": "database-host",
8         "database": "database-name",
9         "user": "database-user",
10        "password": "database-password"
11    }
12 }
```

Figura D.3: Formato del fichero de configuración `project.json`

```
1 server {
2     listen 443 ssl http2;
3     server_name $server_name$;
4     root $route$;
5
6     location / {
7         include proxy_params;
8         proxy_pass http://127.0.0.1:3031;
9     }
10
11    location /socket.io {
12        include proxy_params;
13        proxy_http_version 1.1;
14        proxy_buffering off;
15        proxy_set_header Upgrade
16            $http_upgrade;
17        proxy_set_header Connection "
18            Upgrade";
19        proxy_pass http://127.0.0.1:3031/
20            socket.io;
21    }
22 }
```

Figura D.4: Ejemplo de configuración de servicio en *Nginx*

D.4. COMPILACIÓN, INSTALACIÓN Y EJECUCIÓN DEL PROYECTO

```
1 [Unit]
2 Description=Smartbed Service
3 After=network.target
4 StartLimitIntervalSec=0
5
6 [Service]
7 Type=simple
8 Restart=always
9 RestartSec=1
10 User=http
11 WorkingDirectory=/ruta/a/la/carpeta
12 ExecStart=python index.py
13
14 [Install]
15 WantedBy=multi-user.target
```

Figura D.5: Fichero de `systemd` para ejecutar el proyecto como un servicio

```
1#!/bin/bash
2 /opt/miniconda3/envs/entorno/bin/python index.py
```

Figura D.6: Código para ejecutar un *script Python* en un entorno

esto último no es necesario para el funcionamiento, solo como referencia interna.

Por último es necesario crear un demonio de `systemd` que ejecute la aplicación en segundo plano. El fichero de configuración de este demonio se alojaría en `/usr/lib/systemd/system/smartbeds.service`. Siendo el fichero en cuestión semejante al de la figura D.5:

El campo `ExecStart` podría cambiar en el caso de utilizar algún entornos *Python* como son los que provee conda. En el caso de ser así se recomienda cambiar el campo por `bash index.sh` siendo el fichero semejante al que se puede ver en la figura D.6.

Finalmente solo hace falta lanzar el demonio con el comando `systemctl start smartbeds`.

APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN

ID	Transformer	Test	Resultado	Tiempo (ms)
TB-1	Butter	test_bad_data	OK	
TB-2		test_calculate_dataframe	OK	32
TB-3		test_calculate_series	OK	
TSG-1	SavGol	test_bad_data	OK	
TSG-2		test_calculate_dataframe	OK	24
TSG-3		test_calculate_series	OK	
TC-1	Concatenate	test_bad_data	OK	
TC-2		test_calculate_dataframe	OK	28
TC-3		test_calculate_series	OK	
TEN-1	EachNormalizer	test_bad_data	OK	
TEN-2		test_calculate_dataframe	OK	8
TEN-3		test_calculate_series	OK	
TN-1	Normalizer	test_bad_data	OK	
TN-2		test_bad_data	OK	
TN-3		test_calculate_dataframe	OK	16
TN-4		test_calculate_series	OK	
TMT-1	MoveTargets	test_bad_target	OK	
TMT-2		test_end_seizure	OK	
TMT-3		test_half_seizure	OK	
TMT-4		test_only_seizure	OK	12
TMT-5		test_start_seizure	OK	
TMT-6		test_target_not_boolean	OK	
TVT-1	VarianceThreshold	test_variance0	OK	
TVT-2		test_variance05	OK	8
TVT-3		test_calculate_series	OK	
TP-1	Pipeline	test_bad_data	OK	
TP-2		test_calculate_dataframe	OK	28
TP-3		test_calculate_series	OK	

Tabla D.2: Resultado de las pruebas sobre los transformadores

D.5. Pruebas del sistema

Se han realizado tres conjuntos de pruebas. En primer lugar están las pruebas unitarias de los transformadores de los datos que se encuentran en el directorio `/research` en el fichero `/testLauncher.py`. Las pruebas concretas están en el fichero `/research/tests/transformersTest.py`, los resultados de estas pruebas se pueden ver en la tabla D.2.

Otras pruebas son las unitarias sobre la API, esta se ejecuta con el fichero `/tests.py`, cuyo resultado se puede ver en la tabla D.3, y las pruebas de integración sobre la interfaz que están almacenadas en el fichero `/smartbeds/test/test.side` que se abren con el IDE de Selenium y se ejecutan en una versión local del servidor. Los resultados de estos test se puede ver en la tabla D.4.

ID	Nombre	Resultado	Tiempo (ms)
A-1	test-auth	OK	67
A-2	test-bad-auth	OK	62
A-3	test-user-list	OK	52
A-4	test-user-list-bad-token	OK	31
A-5	test-user-list-no-permition	OK	57
A-6	test-user-add	OK	124
A-7	test-user-add-already-exists	OK	88
A-8	test-user-add-no-admin	OK	57
A-9	test-user-add-bad-token	OK	52
A-10	test-user-mod	OK	66
A-11	test-user-mod-user-not-exist	OK	78
A-12	test-user-mod-no-admin	OK	62
A-13	test-user-mod-bad-token	OK	52
A-14	test-user-mod-own-password-error	OK	47
A-15	test-user-mod-own-password	OK	272
A-16	test-user-del	OK	304
A-17	test-user-del-no-admin	OK	67
A-18	test-user-del-bad-token	OK	52
A-19	test-user-del-admin-error	OK	83
A-20	test-user-mod-own-password	OK	68

Tabla D.3: Resultados de los test unitarios sobre la *API*

ID	Nombre	Resultado
I-1	Iniciar sesión - administrador	OK
I-1.0.1	Iniciar sesión erróneo	OK
I-1.1	Administrar camas	OK
I-1.2	Borrar y modificar cama	OK
I-1.3	Gestionar permisos	OK
I-1.4	Administrar usuarios	OK
I-1.5	Desloguearse	OK
I-2	Iniciar sesión como usuario normal	OK

Tabla D.4: Resultados de los tes sobre la interfaz

Apéndice E

Documentación de usuario

E.1. Introducción

En este apartado se va a explicar como se usa la aplicación para los usuarios posibles. Estos se dividen en dos roles, los administradores y los usuarios.

E.2. Requisitos de usuarios

Todos los tipos de usuario requerirán tener un navegador compatible con HTML 5 y conexión a Internet.

E.3. Instalación

Al tratarse de una aplicación web no se requiere una instalación. La instalación del servicio web se explica en el [Instalación en entorno GNU/Linux](#).

E.4. Manual del usuario

En esta sección se explicarán como usar las distintas funcionalidades de la aplicación web.

Identificación del usuario

El primer paso para utilizar *SmartBeds*, en adelante "la web", es **iniciar la sesión** para acceder a las funcionalidades de la misma.

SmartBeds

Sistema de monitorización de pacientes

Antes de comenzar, por favor, [inicie sesión](#)

Figura E.1: Manual: Ventana inicial.

Iniciar sesión

Nombre de usuario

Contraseña

[Entrar](#)

Figura E.2: Manual: Ventana de *login*.

Todos los pasos del proceso se pueden ver en las figuras E.1 y E.2 :

1. Hacer clic en *inicie sesión*.
2. Introducir su nombre de usuario en el campo *Nombre de usuario* y la contraseña en el campo *Contraseña*.
3. Hacer clic en *Entrar*).

En el caso de que no posea un usuario, o no disponga de su contraseña, contacte con su administrador.

Tras iniciar sesión, si es un usuario del tipo administrador, ve a la sección **Usuario administrador**; en el caso de ser un usuario normal ve a la sección de **Usuario normal**.

Usuario administrador

En este apartado se encuentran las instrucciones para todas las funciones especiales del administrador. Las tareas comunes con el usuario normal se especifican en la sección E.4.

La **barra de navegación** es diferente para los administradores (Fig. E.3) incorporando los hiperenlaces a las tareas de administración.



Figura E.3: Manual: Barra de navegación de administrador.

Administrador camas

Añadir cama		Permisos
Nombre		Acción
Cama 1		<button>Modificar</button> <button>Borrar</button>
Cama 2		<button>Modificar</button> <button>Borrar</button>

Figura E.4: Manual: Administración de las camas.

Administración de camas

Las tareas que se pueden realizar en la administración de camas son la **creación, modificación y eliminación** de camas así como la **gestión de permisos** relacionando la visibilidad de cada cama con cada usuario. Para poder realizar esta administración se debe hacer clic al enlace *Administrar camas* de la barra de navegación (Fig. E.3). La ventana de gestión de las camas se puede ver en la figura E.4.

Los pasos para la **creación de una cama** son los siguientes:

1. Hacer clic en *Añadir cama*.
2. Rellenar los campos del formulario como se puede ver en la figura E.5.
Hay que tener en cuenta que:
 - Todos los campos son obligatorios.
 - La ip ha de ser MULTICAST, es decir estar entre 224.0.0.0 y 239.255.255.255.
 - La dirección MAC y UUID han de ser ambos un valor hexadecimal sin ningún símbolo adicional (guiones, dos puntos etc.).
3. Solamente cuando esté bien cumplimentado se podrá hacer clic en *Guardar cambios*.

Los pasos para **modificar** los datos de una cama son:

Nueva cama ×

Nombre de la cama

Nombre	<input type="text" value="Cama 3"/> ✓
Correcto	

Dirección IP de multidifusión

IP	<input type="text" value="224.0.0.1"/> ✓
Correcto	

Puerto

Puerto	<input type="text" value="600"/> ✓
Correcto	

Dirección MAC

MAC	<input type="text" value="AFEDBA456421"/> ✓
Correcto	

Identificador de cama

UUID	<input type="text" value="BD EDBA564421"/> ✓
Correcto	

Guardar cambios Cerrar

Figura E.5: Manual: Crear cama.

Modificar la cama: Cama 1 ×

Nombre de la cama

Nombre	Cama 1	
Correcto		

Dirección IP de multidifusión

IP	224.3.29.71	
Correcto		

Puerto

Puerto	5007	
Correcto		

Dirección MAC

MAC	886B0F59968A	
Correcto		

Identificador de cama

UUID	00AEFAADD6C3	
Correcto		

Guardar cambios Cerrar

Figura E.6: Manual: Modificar cama.

1. Hacer clic en el botón *Modificar* de la cama deseada.
2. Modificar los datos según se necesite, un ejemplo es la figura E.6. La validación del formulario es igual que en la creación.
3. Hacer clic en *Guardar cambios*.

Para **borrar** una cama se necesitan los siguientes pasos:

1. Hacer clic en *Borrar* de la cama que se deseé.



Figura E.7: Manual: Error al crear, modificar o borrar una cama.

#	alicia	joselu
Cama 1	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Cama 2	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Cerrar

Figura E.8: Manual: Tabla de permisos por usuario.

Estas tres funcionalidades, debido a que no se disponen de camas físicas reales que se puedan escuchar, se han limitado para esta versión, por tanto, cualquiera que sea el proceso realizado no se completará ya que se recibirá el error de la figura E.7.

La **modificación de los permisos** se realiza con los siguientes pasos:

1. Hacer clic en *Permisos*.
2. Hacer clic en las cajas según se quiera dar o quitar permisos (Fig. E.8).

Administración de usuarios

Las tareas que se pueden realizar en la administración de usuarios son la **creación, modificación de contraseña y eliminación** de usuarios. Para

Administrar usuarios

Añadir usuarios	
Nombre	Acción
admin	<button>Modificar</button>
alicia	<button>Modificar</button> <button>Borrar</button>
joselu	<button>Modificar</button> <button>Borrar</button>

Figura E.9: Manual: Administración de los usuarios.

poder realizar esta administración se debe hacer clic al enlace *Administrar usuarios* de la barra de navegación (Fig. E.3). La ventana de gestión de las camas se puede ver en la figura E.9

Los pasos para la **creación de un usuario** son:

1. Hacer clic en *Añadir usuarios*.
2. Rellenar el formulario (Fig. E.10), el *nickname* ha de tener de tamaño máximo 50 caracteres.
3. Hacer clic en *Guardar cambios*.

Los pasos para **modificar la contraseña de un usuario** son:

1. Hacer clic en *Modificar* del usuario deseado.
2. Rellenar el formulario (Fig. E.11).
3. Hacer clic en *Guardar cambios*

Los pasos para **borrar un usuario** son:

1. Hacer clic en *Borrar* del usuario deseado.

Usuario normal

La primera página que un usuario que se ha identificado es la lista de camas disponibles y su estado (Fig. E.12). Haciendo clic en cualquiera de los botones accederá a la página de esa cama (Fig. E.13). En la parte izquierda puede ver las constantes vitales, estos son:

Nuevo usuario ×

Nickname del usuario

Nickname	prueba	
Correcto		

Contraseña

Contraseña	*****	
Correcto		

Repetir contraseña	*****	
Correcto		

Guardar cambios Cerrar

Figura E.10: Manual: Formulario de creación de usuario.

Modificar contraseña de: alicia ×

Nickname del usuario

Nickname	alicia	
Correcto		

Contraseña

Contraseña	*****	
Correcto		

Repetir contraseña	*****	
Correcto		

Guardar cambios Cerrar

Figura E.11: Manual: Formulario de modificación de usuario.



Figura E.12: Manual: Formulario de modificación de usuario.

- **HR:** *Hearth rate*, frecuencia cardíaca en pulsaciones por minuto.
- **RR:** *Respiration rate*, frecuencia respiratorias en respiraciones por minuto.
- **SV:** *Stroke volume*, volumen sistólico en mililitros.
- **HRV:** *Heart rate variability*, variabilidad de la frecuencia cardíaca en milisegundos.
- **B2B:** *Betweenn two beats*, tiempo entre dos pulsaciones en milisegundos.

En la parte derecha se encuentran los valores de las presiones de los seis tubos de la cama. Las posiciones de estos tubos son:

- **P1:** Cabeza.
- **P2:** Hombros.
- **P3:** Abdomen.
- **P4:** Cadera.
- **P5:** Rodillas.
- **P6:** Pies.

La parte más importante es el **estado** del paciente, este puede ser dormido (Fig. E.14b), en una crisis (Fig. E.14c) o despierto (Fig. E.14a). En las dos primeras situaciones incluye la probabilidad de una situación de crisis.

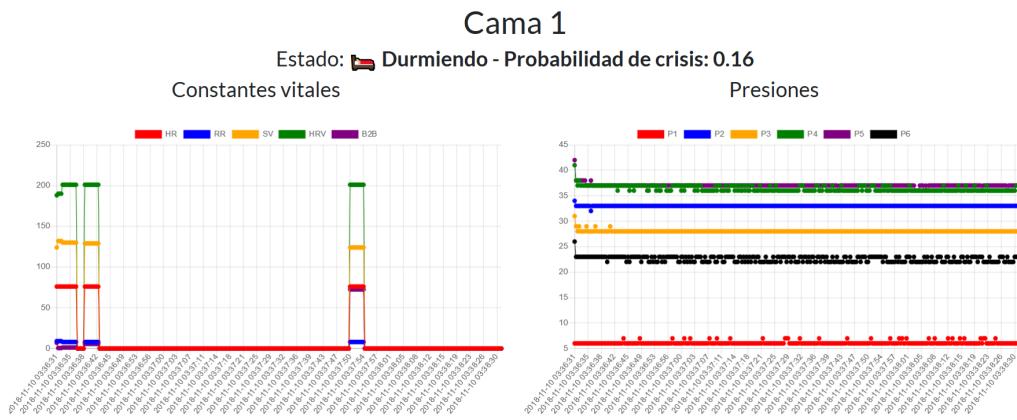


Figura E.13: Manual: Página con la información en tiempo real de una cama.

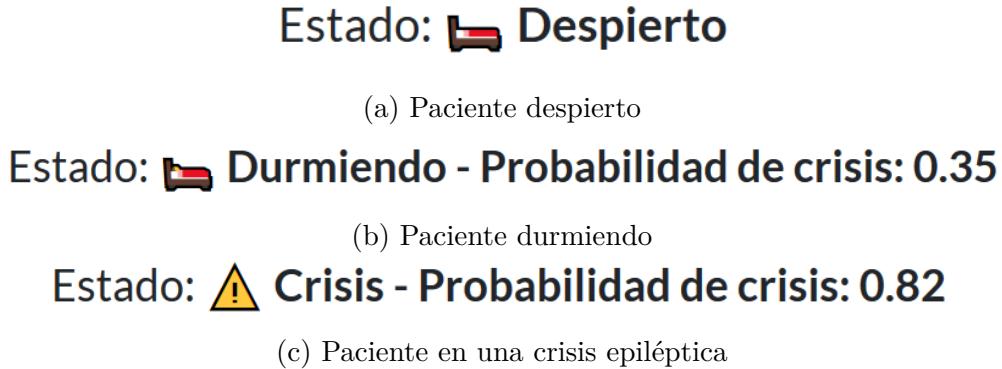


Figura E.14: Manual: diferentes estados de un paciente



Figura E.15: Manual: Barra de navegación, lado derecho.

Cambiar la contraseña

Para cambiar la contraseña de su usuario se necesita hacer clic al nombre del usuario en la parte derecha (Fig. E.15) y llenar el formulario al que se redirige (Fig. E.16).

Cambiar contraseña

Contraseña actual:	<input type="text" value="Contraseña actual"/>
Contraseña nueva:	<input type="text" value="Nueva contraseña"/>
Repite la nueva contraseña	<input type="text" value="Repite la nueva contraseña"/>
<input type="button" value="Actualizar contraseña"/>	

Figura E.16: Manual: Formulario de cambio de contraseña.

Bibliografía

- [1] Damien Arrachequesne and Erik Little. Socket.io. <https://socket.io/docs/>, feb 2018.
- [2] R. Fielding and J. Reschke. Hypertext transfer protocol (http/1.1): Semantics and content. RFC 7231, RFC Editor, June 2014. <http://www.rfc-editor.org/rfc/rfc7231.txt>.
- [3] Indeed. Salarios para empleos de a.c.s. informáticos en España. <https://www.indeed.es/cmp/A.c.s.-Inform%C3%A1ticos/salaries>, jun 2019.
- [4] Wikipedia contributors. Model–view–controller — Wikipedia, the free encyclopedia. <https://en.wikipedia.org/w/index.php?title=Model%E2%80%93view%E2%80%93controller&oldid=901347868>, 2019. [Online; accessed 11-June-2019].
- [5] Wikipedia contributors. Responsive web design — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Responsive_web_design&oldid=899574617, 2019. [Online; accessed 31-May-2019].
- [6] Wikipedia contributors. Websocket — Wikipedia, the free encyclopedia. <https://en.wikipedia.org/w/index.php?title=WebSocket&oldid=897985416>, 2019. [Online; accessed 31-May-2019].
- [7] Zhenyue Zhang and Hongyuan Zha. Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. *SIAM journal on scientific computing*, 26(1):313–338, 2004.



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería
Informática**

SmartBeds

**Apéndice: Cuaderno de
investigación**



Presentado por José Luis Garrido Labrador
y Alicia Olivares Gil
en Universidad de Burgos – 25 de junio
de 2019

Tutores: Dr. Álvar Arnaiz González
y Dr. José Francisco Díez Pastor

Índice general

Índice general	I
Índice de figuras	IV
Índice de tablas	VI
1 Descripción de los datos	1
1.1. Introducción	1
1.2. Representación de las señales de presión	2
2 Primeros pasos	5
2.1. Introducción	5
2.2. Técnicas empleadas	5
3 Transformadores	9
3.1. Introducción	9
3.2. Preprocesado	9
3.3. Estadísticos	9
3.4. Filtrado	10
3.5. Compuestos	10
3.6. De instancias	10
4 Análisis de componentes principales (PCA)	11
4.1. Introducción	11
4.2. Preprocesamiento	11
4.3. Entrenamiento y testeo	11
5 Proyecciones a 2 dimensiones	15

5.1. Introducción	15
5.2. Preprocesamiento	15
5.3. Isomap	16
5.4. Locally Linear Embedding (LLE)	16
5.5. Multi-dimensional Scaling (MDS)	17
5.6. Otros métodos	17
6 One-Class simple	23
6.1. Introducción	23
6.2. Configuración	23
6.3. Preprocesado	23
6.4. Entrenamiento	24
6.5. Resultados	24
7 Ensembles para desequilibrados	29
7.1. Introducción	29
7.2. Resultados	29
7.3. Conclusiones	31
8 Random Forest – Media y desviación móviles	33
8.1. Introducción	33
8.2. Modificación del target	33
8.3. Validación cruzada entre dos días	34
8.4. Primera exploración, métrica=ROC	34
8.5. Segunda exploración, métrica=ROC	35
8.6. Tercera exploración, métrica=ROC	36
8.7. Primera exploración, métrica=Precision-Recall	37
8.8. Segunda exploración, métrica=Precision-Recall	38
8.9. Tercera exploración, métrica=Precision-Recall	38
9 Extracción de características con <i>tsfresh</i> y clasificador Random Forest	41
9.1. Introducción	41
9.2. Filtrado típico de características	41
9.3. Filtrado mediante la función <code>select_features</code> de <i>tsfresh</i>	42
9.4. Selección del mejor conjunto de características	43
10 Ensembles para desequilibrados <i>tsfresh</i>	51
10.1. Introducción	51
10.2. Resultado de experimentos	52
10.3. Comentario de los resultados	52

ÍNDICE GENERAL

III

10.4. Conclusiones 55

Bibliografía 57

Índice de figuras

1.1. Señales de los datos brutos y estadísticos asociados al tubo de presión P5. Estimación de la crisis epiléptica en rojo	3
2.1. Proporción de datos ruidos por tubo de presión	7
2.2. Filtrado de datos	8
4.1. PCA ajustada a cada crisis	12
4.2. PCA ajustada con todas las crisis	13
5.1. Proyección Isomap 2D de los datos brutos de presiones. Crisis epiléptica en rojo.	16
5.2. Proyección Isomap 2D de los datos estadísticos de presiones. Crisis epiléptica en rojo.	17
5.3. Proyección LLE 2D de los datos brutos de presiones. Crisis epiléptica en rojo.	18
5.4. Proyección LLE 2D de los datos estadísticos de presiones. Crisis epiléptica en rojo.	19
5.5. Proyección MDS 2D de los datos brutos de presiones. Crisis epiléptica en rojo.	20
5.6. Proyección MDS 2D de los datos estadísticos de presiones. Crisis epiléptica en rojo.	21
8.1. Mapa de calor de las áreas bajo la curva calculadas en la primera exploración con la métrica=ROC.	35
8.2. Mapa de calor de las áreas bajo la curva calculadas en la segunda exploración con la métrica=ROC.	36
8.3. Mapa de calor de las áreas bajo la curva calculadas en la tercera exploración con la métrica=ROC.	37

Índice de figuras

v

8.4. Mapa de calor de las áreas bajo la curva calculadas en la primera exploración con la métrica=Precision-Recall.	38
8.5. Mapa de calor de las áreas bajo la curva calculadas en la segunda exploración con la métrica=Precision-Recall.	39
8.6. Mapa de calor de las áreas bajo la curva calculadas en la tercera exploración con la métrica=Precision-Recall.	39
9.1. Gráficas de la evolución de las poblaciones en el algoritmo genético con la métrica=ROC.	48
9.2. Gráficas de la evolución de las poblaciones en el algoritmo genético con la métrica=Precision-Recall.	49
10.1. Métodos para desbalanceados - Entrenamiento con la primera crisis, testeo con la segunda crisis.	53
10.2. Métodos para desbalanceados - Entrenamiento con la segunda crisis, testeo con la primera crisis.	54

Índice de tablas

6.1. Matriz de confusión entrenamiento con 1º crisis	25
6.2. Matriz de confusión entrenamiento con 1º y 2º crisis	25
6.3. Matriz de confusión test con 2º crisis (entrenamiento con 1ª)	26
6.4. Matriz de confusión test 3º crisis (entrenamiento 1º)	27
6.5. Matriz de confusión test 3º crisis (entrenamiento con 1º y 2º)	27
7.1. Matrices de confusión - SMOTE Train-Test	30
7.2. Matrices de confusión - SMOTE CrossValidation	30
10.1. Métodos para desbalanceados - Entrenamiento con la primera crisis, testeo con la segunda crisis.	52
10.2. Métodos para desbalanceados - Entrenamiento con la segunda crisis, testeo con la primera crisis.	54

Capítulo 1

Descripción de los datos

1.1. Introducción

Los datos con los que vamos a trabajar provienen de fichero de extensión *CSV* con las siguientes columnas:

- MAC_NGMATT: Es el identificador del colchón, asociado a los tubos de presión.
- UUID_BSN: Es el identificador del sensor de ritmo cardíaco y respiración instalado en el interior del colchón.
- DateTime: Día y hora en la que se toma cada instancia de dato. Tenemos aproximadamente entre 1 y 3 instancias de dato por segundo.
- P1, P2, P3, P4, P5, P6, P7, P8, P9, P10, P11, P12: Presiones, en mBar, captadas por los 12 tubos de presión alojados dentro del colchón. En nuestro caso solo tendremos información en los primeros 6 campos (de P1 a P6) ya que se trata de un colchón individual.
- SS: Potencia de la señal relativa a los tubos de presión. Un valor superior a 400 se considera aceptable. En el preprocesado no tendremos en cuenta las instancias de datos con valores de SS menores.
- HR: Ritmo cardíaco expresado en pulsaciones/minuto.
- RR: Ritmo respiratorio expresado en respiraciones/minuto.
- SV: Volumen sistólico expresado en mililitros.

- HRV: Variabilidad del ritmo cardíaco expresado en ms.
- B2B: Tiempo entre pulsaciones expresado en ms.
- STATUS: Parámetro que identifica el estado de medición del sensor de ritmo cardíaco y respiración. 0=*low signal*, 1=*ok signal*, 2=*high signal*, 3=*close to overload*, 4=*close to max HR*.

1.2. Representación de las señales de presión

En la Figura: 1.1 representamos un ejemplo de señal de presión en el tiempo junto con las señales de sus estadísticas móviles (media, desviación y rango) usando una ventana de tamaño 25.

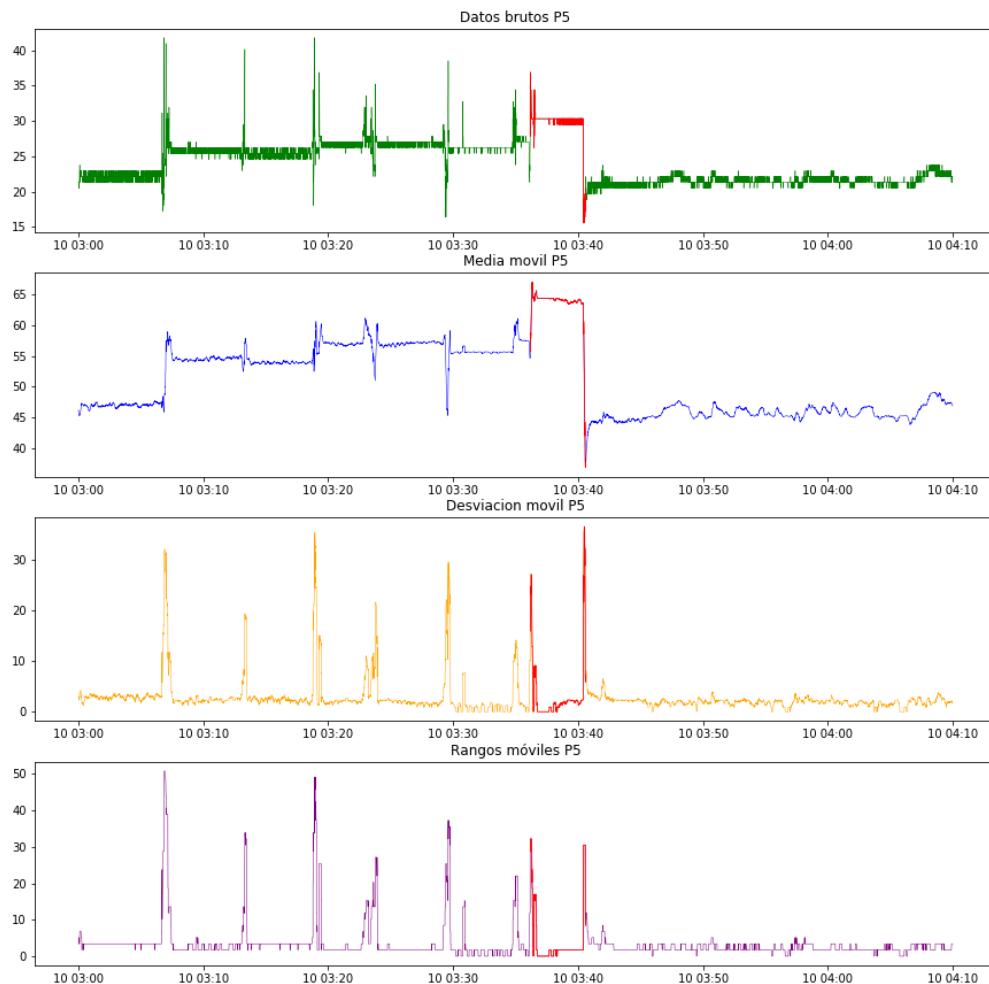


Figura 1.1: Señales de los datos brutos y estadísticos asociados al tubo de presión P5. Estimación de la crisis epiléptica en rojo.

Capítulo 2

Primeros pasos

2.1. Introducción

En cuanto al trabajo hecho hasta el momento, vamos a hablar del preprocesado de los datos, incluyendo la carga de datos, la selección de los datos útiles y el cálculo estadísticas móviles (media y varianza). Hablamos también de las técnicas de reducción de dimensionalidad de los datos que hemos explorado, y cuáles hemos seleccionado, y algunas técnicas de filtrado.

2.2. Técnicas empleadas

- Eliminación por baja señal y de datos erróneos
- Eliminación manual de ruido
- Normalización
- Filtrados
- Eliminación de baja variabilidad
- Cálculo de estadísticas móviles

Eliminación por baja señal

Nada más comenzar¹, y para todos los casos, hemos eliminado los datos que tienen por señal 0, (baja calidad) y todos los valores que no sean presiones también son eliminados debido a que en la mayor parte de las ocasiones dan valores nulos de manera intermitente.

Eliminación manual de ruido

Observamos que en algunas ocasiones había datos aislados y valores muy pequeños o incluso negativos. Por ende, decidimos considerar todo valor menor de 5 como ruido y lo convertimos a 0. Con esta medida eliminamos también los valores negativos.

Según este criterio, podemos ver que algunos tubos como son el de la cabeza, las rodillas y los pies (figuras 2.1a, 2.1e y 2.1f) son especialmente ruidosos debido a la baja sensibilidad de los tubos. En el resto de tubos (figuras 2.1b, 2.1c y 2.1d) podemos ver que las proporciones son semejantes, que suceden en su mayoría en los momentos en los que la cama está vacía².

Normalización

Para trabajar siempre de la misma forma se normalizaron todos los datos, de 0 a 100, respecto a la presión

Filtrados

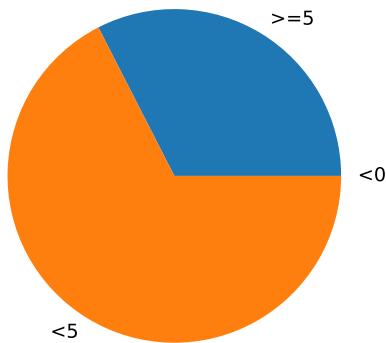
El filtrado es otra técnica empleada para eliminar el ruido, hemos empleado varias filtros como el *butterworth* [16] con valores $N=3$ y $Wn=0.05$ cuyo resultado se puede ver en la Figura 2.2a. Independientemente del tipo de filtro se aplican con la función `scipy.signal.filtfilt`. También se han probado el filtro *Savitzky–Golay* [18] que se aplica con `scipy.signal.savgol_filter` directamente el el resultado se puede ver en la Figura 2.2b.

Eliminación de baja variabilidad

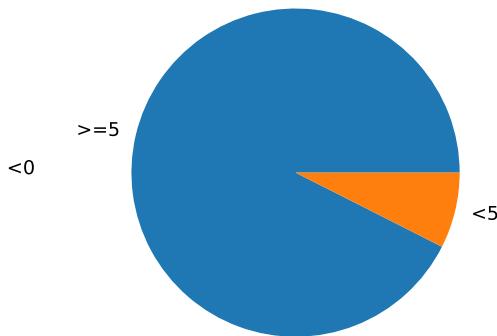
Para hacer más pequeño el conjunto de datos eliminamos aquellos tubos cuya variabilidad sea menor a un umbral, por defecto ese umbral de de 0.5.

¹Se aplica independientemente de que otras técnicas de procesamiento se empleen, la salida de esto se considera **bruta**

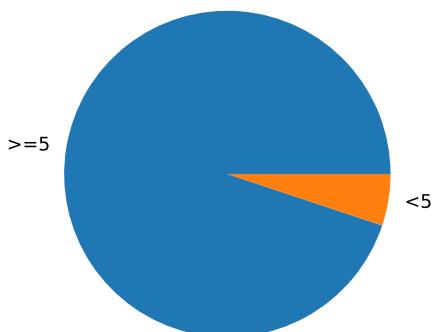
²Este ha sido el criterio para determinar al valor 5 como el mínimo para un valor real de presión



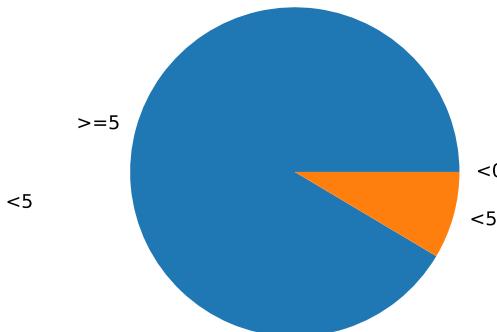
(a) Proporción de ruido en tubo de presión 1



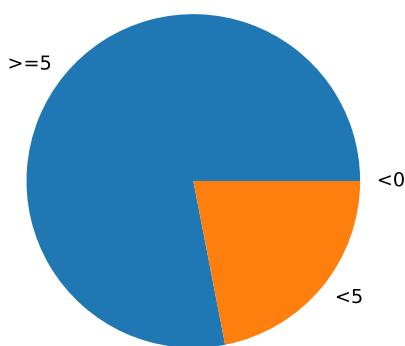
(b) Proporción de ruido en tubo de presión 2



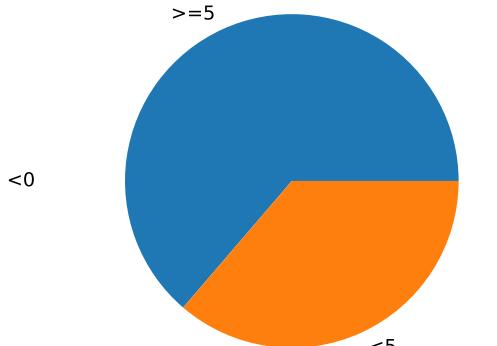
(c) Proporción de ruido en tubo de presión 3



(d) Proporción de ruido en tubo de presión 4

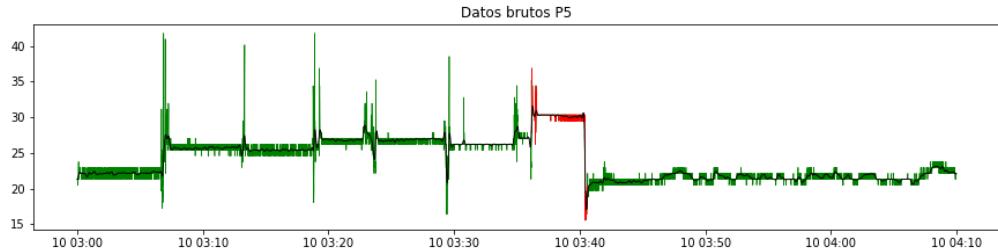


(e) Proporción de ruido en tubo de presión 5

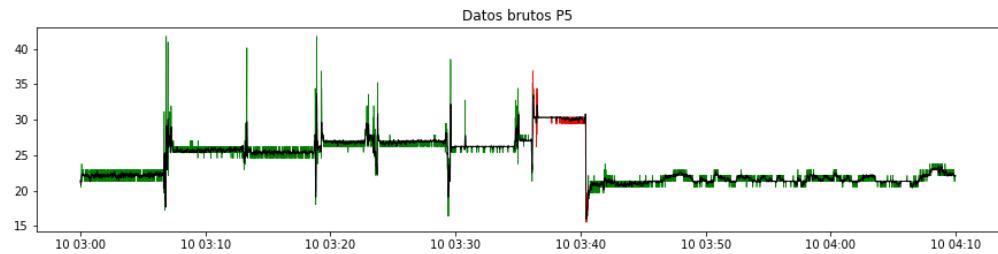


(f) Proporción de ruido en tubo de presión 6

Figura 2.1: Proporcion de datos ruidos por tubo de presión



(a) Ejemplo de señal de presión filtrada mediante *butterworth* con $N=3$ y $Wn=0.05$. En verde la señal original y en negro la señal filtrada.



(b) Ejemplo de señal de presión filtrada mediante *Savitzky–Golay* con `window_length=15` y `polyorder=2`. En verde la señal original y en negro la señal filtrada.

Figura 2.2: Filtrado de datos

Cálculo de estadísticas móviles

A partir de los datos calculamos la media móvil y la varianza móvil con una ventana de tamaño 25. El tamaño se ha escogido como aproximación inicial y se cambiará para futuras pruebas.

Capítulo 3

Transformadores

3.1. Introducción

Hemos agrupado las distintas partes del preprocesado en *Transformers* compatibles con `sklearn` [6]. Los hemos dividido en varios grupos.

3.2. Preprocesado

Los transformadores de preprocesado son aquellos pensados para las operaciones básicas sobre los datos para facilitar la ejecución de otras más complejas. Las desarrolladas han sido:

- `Normalizer` para la normalización de los datos entre 0 y 1
- `NoiseFilter(minimum=0)` transforma en 0 para valores menores del `minimum`
- `VarianceThresholdPD(threshold=0.05)` elimina los datos con una varianza menor de 0,05 compatible con `pandas` al mantener los mismos nombres

3.3. Estadísticos

Para facilitar el procesado de estadísticos (media, desviación, rango, máximo y mínimo) hemos desarrollado un transformador genérico cuyos modos pueden ser `mean`, `std`, `range`, `max`, `min` y calcula una operación

móvil sobre los datos. La firma del método sería: `StatisticsTransformer(mode='mean', window=25)`.

3.4. Filtrado

Para el filtrado se han creado dos transformadores para dos filtros, *ButterWorth* y *Savitzky-Golay* aunque se pueden crear más extendiendo de `FilterTransformer()`. La firma de los métodos sería:

- `ButterTransformer(N, Wn, btype='low', analog=False, output='ba', fs=None)`
- `SavgolTransformer(window_length, polyorder=2, deriv=0, delta=1.0, axis=-1, mode='interp', cval=0.0)`

3.5. Compuestos

Para agrupar distintos transformadores se han creado dos según la operación que se desea realizar, ya sea una concatenación (`ConcatenateTransformer`) y la aplicación en serie de transformadores (`PipelineTransformer`). Ambos reciben en sus constructores un conjunto indefinido de transformadores a aplicar en el orden deseado para aplicarlos.

3.6. De instancias

Los transformadores de instancias son aquellos que modifican completamente las instancias, incluyendo las fechas y las etiquetas. En particular el que existe en `MoveTargetsTransformer` que cambia los *targets* de posición según una ventana para adaptar correctamente las ventanas según los valores de los estadísticos móviles.

Tiene cuatro modos, `only` que únicamente marca como crisis si el estadístico ha sido realizado solamente con datos de crisis, `half` donde se marca como crisis si al menos la mitad de los datos con los que se han realizado la estadística son crisis, `start` que marca como crisis si el primer elemento de los datos es una crisis y `end` que realiza lo contrario, si el último elemento es crisis¹

¹Los datos cuando son procesados con un estadístico cualquiera ya son de esta manera

Capítulo 4

Análisis de componentes principales (PCA)

4.1. Introducción

Para comprobar si los datos de las presiones eran separables de manera fácil utilizamos PCA con los datos de las tres crisis documentadas a fecha de 2019-02-15.

4.2. Preprocesamiento

El preprocesamiento utilizado ha sido el filtro de ruido a 5, la normalización de los datos entre 0 y 100 y la eliminación de tubos con una varianza menor a 0.5. Además escogemos los datos que tengan un valor de SS mayor que 4000.

4.3. Entrenamiento y testeo

El análisis con la primera crisis Fig. 4.1a tuvo de resultado que la crisis (rojo) no era separable de las situaciones normales (azul), además tampoco compartía espacio con las otras crisis (amarillo):

De manera semejante ocurre ajustando con la segunda crisis y la tercera Fig. 4.1b, 4.1c

Si hacemos un ajuste con los datos de las tres crisis podemos ver que no son separables las situaciones de crisis y los datos normales Fig. 4.2

10 CAPÍTULO 4. ANÁLISIS DE COMPONENTES PRINCIPALES (PCA)

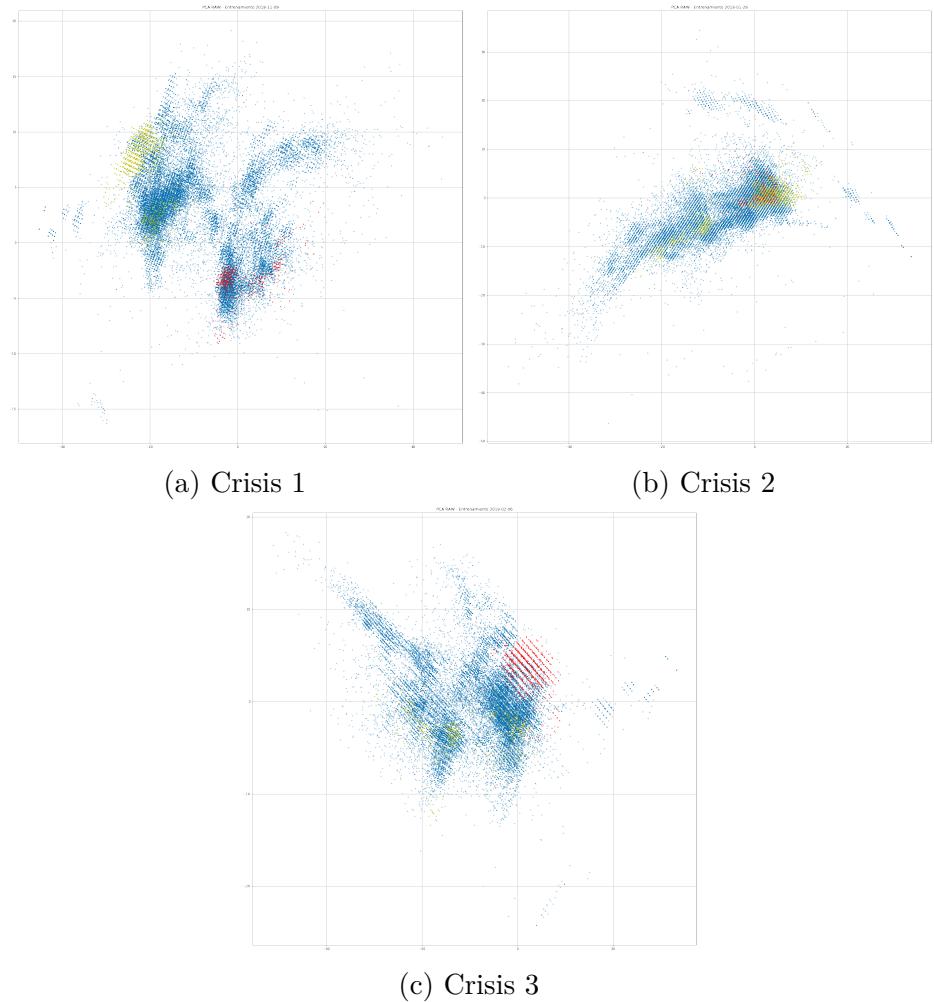


Figura 4.1: PCA ajustada a cada crisis

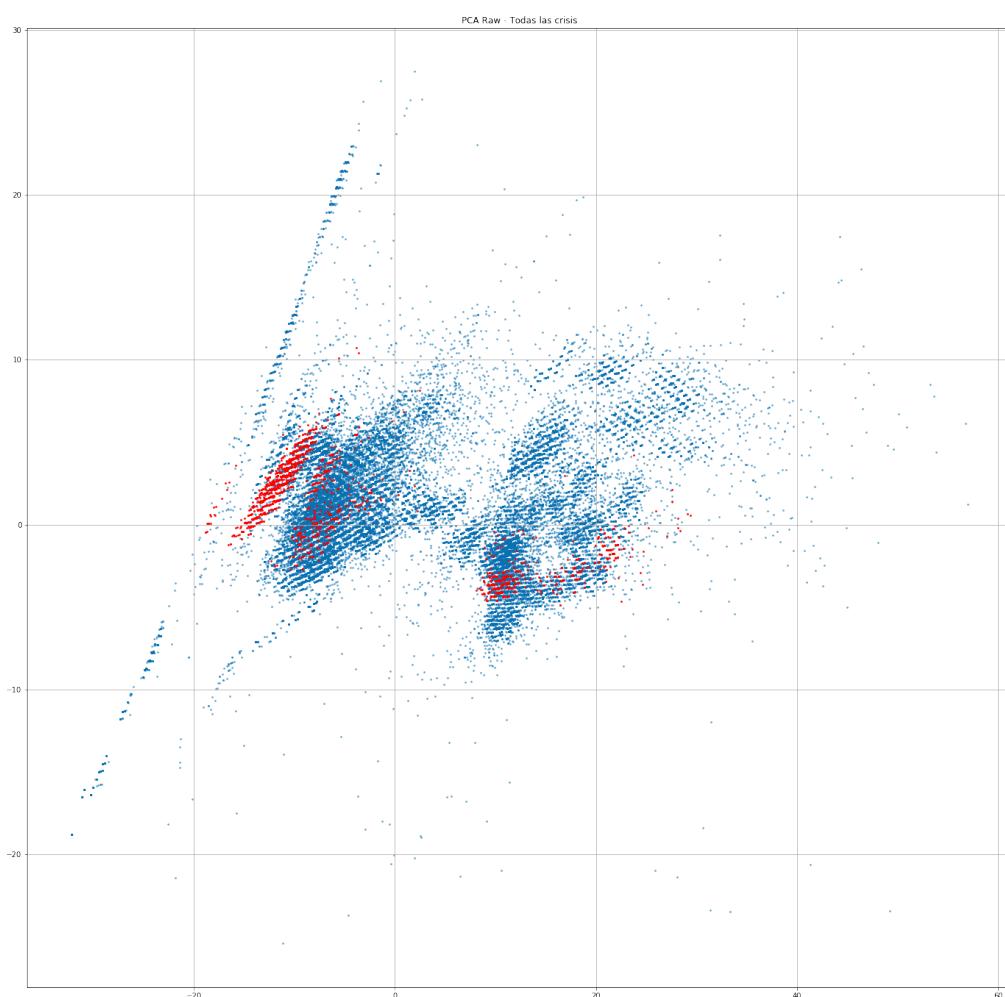


Figura 4.2: PCA ajustada con todas las crisis

Capítulo 5

Proyecciones a 2 dimensiones

5.1. Introducción

También hemos aplicado las implementaciones de `sklearn.manifold` con los valores de presión correspondiente a la crisis que tuvo lugar el día 2018-11-10. El objetivo es comprobar si las crisis son fácilmente separables del resto de datos al realizar su proyección en un espacio de 2 dimensiones con alguna de las técnicas incluidas en la librería. A diferencia de PCA, estas implementaciones no permiten proyectar nuevos datos a partir de un modelo ya entrenado, por lo que la representación de las proyecciones solo incluirá los datos con los que se ha entrenado el modelo. Esto impide que estas proyecciones puedan ser usadas para extraer características.

Es importante destacar que el volumen de datos que admiten estos métodos es limitado para los recursos que tenemos, lo que no nos permite entrenar los modelos con más de una crisis. Los parámetros de los métodos se han escogido teniendo en cuenta esta limitación.

5.2. Preprocesamiento

Antes de aplicar las distintas técnicas se han realizado las siguientes operaciones:

- Selección de los datos comprendidos entre 20 minutos antes del inicio de la crisis y 20 minutos después de su finalización.
- Filtro de ruido a 5 y normalización de los datos brutos entre 0 y 100.

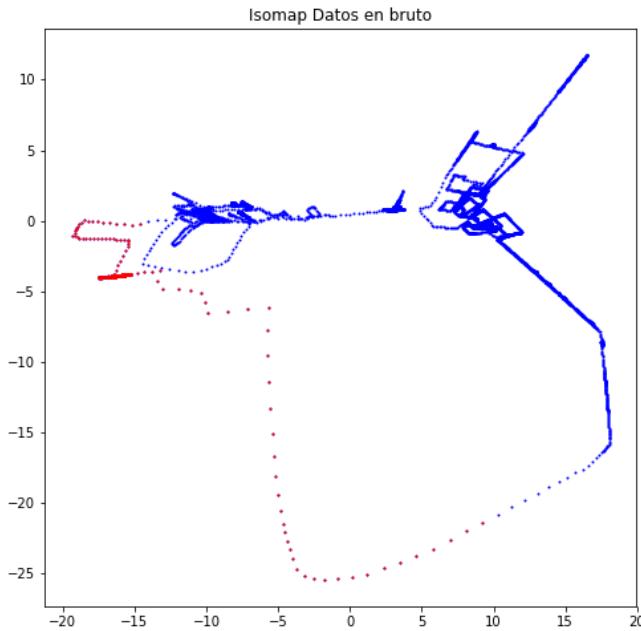


Figura 5.1: Proyección Isomap 2D de los datos brutos de presiones. Crisis epiléptica en rojo.

- Cálculo de media, desviación y rangos móviles con ventana 25.
- Normalización de los datos de media, desviación, y rango por separado para aplicar las proyecciones sobre ellos conjuntamente.
- Filtro butterworth $N=3$ y $Wn=0.05$ tanto a los datos brutos como a los datos estadísticos.

5.3. Isomap

Aplicamos Isomap [14] con `n_neighbors=10` y `n_components=2` a los datos en bruto (ver Fig. 5.1) y a los datos estadísticos (ver Fig. 5.2).

5.4. Locally Linear Embedding (LLE)

Aplicamos LLE [12] con `n_neighbors=10` y `n_components=2` a los datos en bruto (ver Fig. 5.3) y a los datos estadísticos (ver Fig. 5.4).

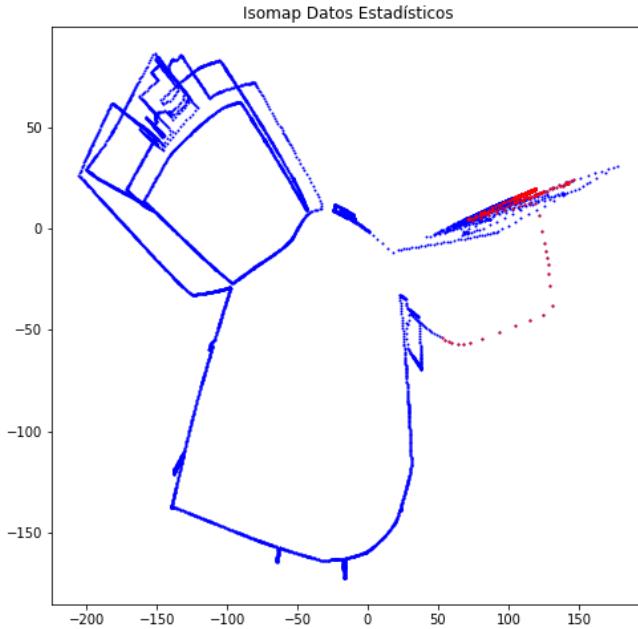


Figura 5.2: Proyección Isomap 2D de los datos estadísticos de presiones. Crisis epiléptica en rojo.

5.5. Multi-dimensional Scaling (MDS)

Aplicamos MDS [10, 1] con `n_components=2` y `max_iter=100` a los datos en bruto (ver Fig. 5.5) y a los datos estadísticos (ver Fig. 5.6).

5.6. Otros métodos

Se han empleado otros métodos cuyas proyecciones no merecen ser consideradas al no presentar ningún tipo de separación aparente entre los datos de la crisis y el resto.

- Modified Locally Linear Embedding (MLLE [19]) con `n_neighbors=15` y `n_components=2`.
- Hessian Eigenmapping con `n_neighbors=20` y `n_components=2`.

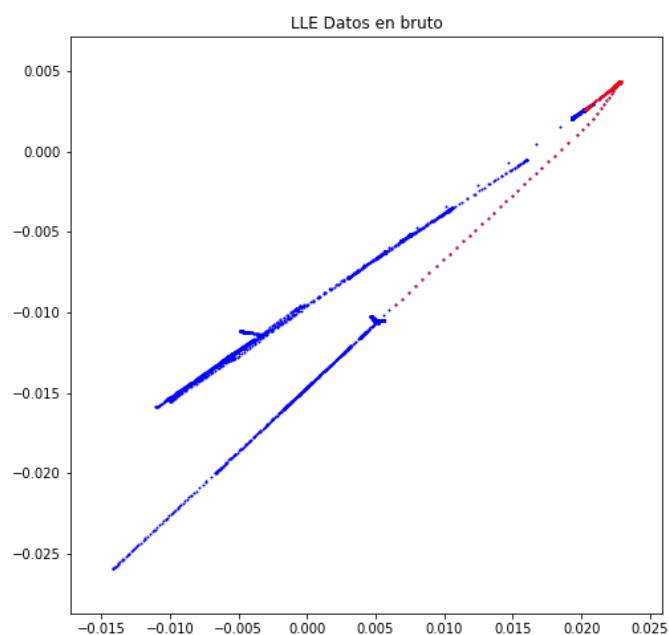


Figura 5.3: Proyección LLE 2D de los datos brutos de presiones. Crisis epiléptica en rojo.

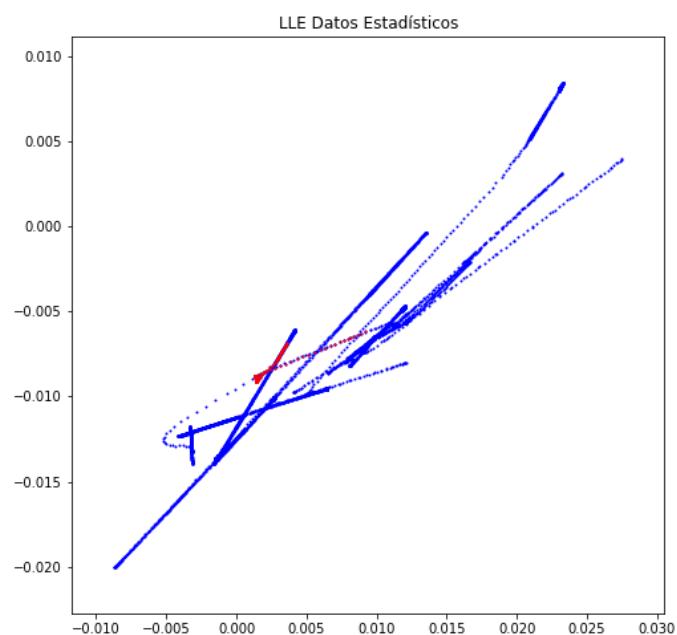


Figura 5.4: Proyección LLE 2D de los datos estadísticos de presiones. Crisis epiléptica en rojo.

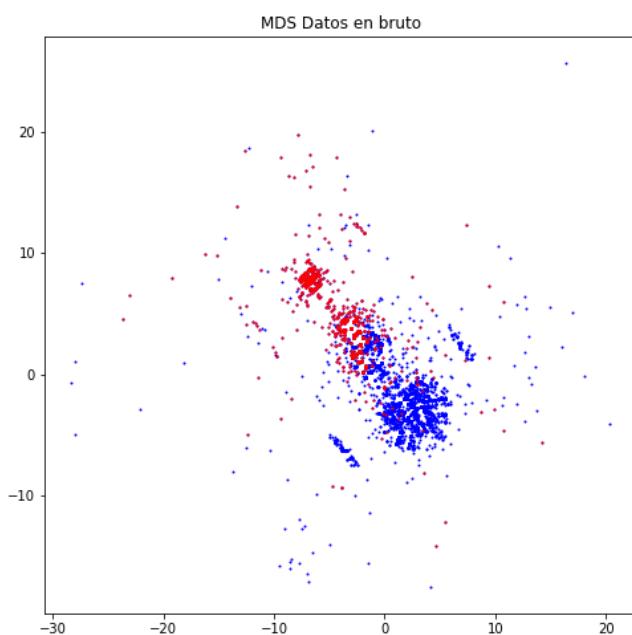


Figura 5.5: Proyección MDS 2D de los datos brutos de presiones. Crisis epiléptica en rojo.

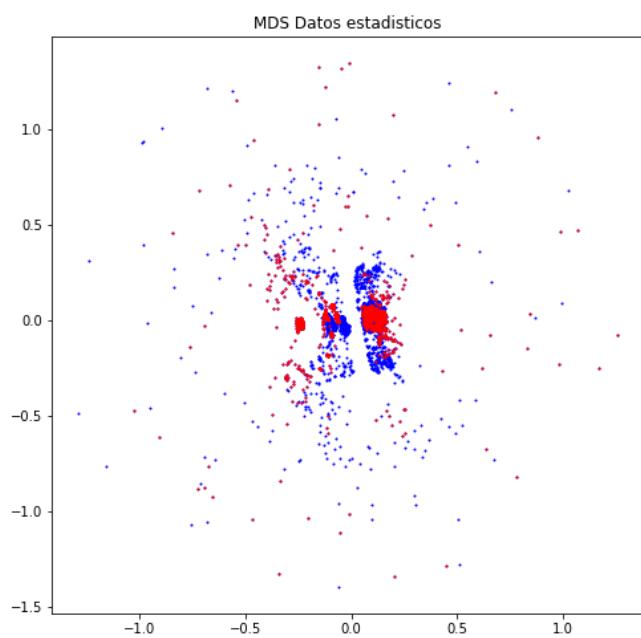


Figura 5.6: Proyección MDS 2D de los datos estadísticos de presiones. Crisis epiléptica en rojo.

Capítulo 6

One-Class simple

6.1. Introducción

Debido a que el problema que intentamos resolver es desequilibrado [8] una línea que hemos explorado ha sido la detección de anomalías, por tanto se ha probado a entrenar un clasificador de *One-Class* de máquina de vectores soporte (SVM) con sklearn. Para realizar este aprendizaje ignoramos una de las clases por lo que el problema a resolver contiene el máximo desequilibrio ya que existen solo datos de una clase.

6.2. Configuración

La configuración del clasificador ha sido la por defecto teniendo en cuenta un *kernel* de tipo *Radial Basis Function* [15], *gamma* de 0,01 y *nu* de 0,1

6.3. Preprocesado

Se han probado seis configuraciones de preprocesado distintas, tres bases y las equivalentes en estadísticas móviles, estas han sido de ventana 25. A su vez, cada estadística por separado (Media, desviación y rango) ha sido normalizado entre 0 y 100. Además se han eliminado aquellas columnas con una varianza menor a 0.5.¹ Los procesados han sido dos más los valores en bruto². Se han aplicado el filtro `scipy.signal.savgol_filter` con ventana

¹Aunque esto no ha perturbado los datos

²Eliminado ruido en el umbral 5, normalizado entre 0 y 100 y eliminación de tubos con poca varianza menor de 0.5

de 15 y, paralelamente, un filtro `scipy.signal.butter` con los valores 3 y 0,05

6.4. Entrenamiento

El entrenamiento se ha realizado de manera alternativa entre solo crisis o la situación normal con todos los conjuntos de datos puestos con anterioridad. En particular se han usado los datos de la noche entre el 9 y el 10 de noviembre de 2018 que contiene una crisis documentada entre las 03:30:00 y las 3:50:00. Aunque se consideró el inicio real como 03:36:10 porque los valores cambian abruptamente y las 03:40:37 cuando vuelven a cambiar abruptamente.

También probamos a entrenar con la primera y la segunda crisis. Esta que fue el 28 de enero la consideramos entre las 2019-01-29 06:12:04 y las 2019-01-29 06:15:37 al ser el área más anómala de la crisis debido a que la crisis no estaba etiquetada correctamente.

La tercera crisis, fue el 6 de febrero y esta fue etiquetada correctamente.³

6.5. Resultados

Los resultados entrenando con el conjunto de *no-crisis* fueron en una matriz de confusión no diagonalmente dominante, por tanto solamente nos centraremos en la predicción con el entrenamiento de *no-crisis*

Datos de entrenamiento

El entrenamiento se realizó de dos maneras distintas, una utilizando solamente datos de la primera crisis y otra utilizando los datos de las dos primeras crisis. Los resultados se pueden ver en las tablas agrupadas en la Tabla 6.1 para el entrenamiento con una crisis y Tabla 6.2 con las dos primeras crisis

Datos de test

La validación se ha realizado con la segunda y tercera crisis (por separado) para el entrenamiento con una sola crisis y la tercera crisis cuando se ha entrenado con las dos primeras. Los resultados se pueden ver en Tabla 6.3 y Tabla 6.4 para las validaciones con el entrenamiento de una sola crisis y

³Aunque en precisión de minutos y no de segundos.

	No Crisis	Crisis		No Crisis	Crisis
No Crisis	103419	6	No Crisis	103426	0
Crisis	71	553	Crisis	59	540

(a) Datos Brutos	(b) Datos Estadísticos		
No Crisis	Crisis	No Crisis	Crisis
103421	4	103426	0
62	562	Crisis	60 539

(c) Filtrado Savitzky–Golay	(d) Savitzky–Golay Estadístico		
No Crisis	Crisis	No Crisis	Crisis
103425	0	103426	0
62	562	Crisis	61 538

(e) Filtrado ButterWorth	(f) ButterWorth Estadístico		
No Crisis	Crisis	No Crisis	Crisis
171109	36209	207295	0
117	1042	Crisis	122 1012

Tabla 6.1: Matriz de confusión entrenamiento con 1º crisis

	No Crisis	Crisis		No Crisis	Crisis
No Crisis	171109	36209	No Crisis	207295	0
Crisis	117	1042	Crisis	122	1012

(a) Datos Brutos	(b) Datos Estadísticos		
No Crisis	Crisis	No Crisis	Crisis
185987	21331	207295	0
116	1043	Crisis	122 1012

(c) Filtrado Savitzky–Golay	(d) Savitzky–Golay Estadístico		
No Crisis	Crisis	No Crisis	Crisis
192713	14605	207295	0
115	1044	Crisis	164 970

(e) Filtrado ButterWorth	(f) ButterWorth Estadístico		
No Crisis	Crisis	No Crisis	Crisis
171109	36209	207295	0
117	1042	Crisis	122 1012

Tabla 6.2: Matriz de confusión entrenamiento con 1º y 2º crisis

	No Crisis	Crisis		No Crisis	Crisis
No Crisis	103893	0	No Crisis	103894	0
Crisis	535	0	Crisis	510	0
(a) Datos Brutos			(b) Datos Estadísticos		
	No Crisis	Crisis		No Crisis	Crisis
No Crisis	103893	0	No Crisis	103894	0
Crisis	535	0	Crisis	510	0
(c) Filtrado Savitzky–Golay			(d) Savitzky–Golay Estadístico		
	No Crisis	Crisis		No Crisis	Crisis
No Crisis	103893	0	No Crisis	103894	0
Crisis	535	0	Crisis	510	0
(e) Filtrado ButterWorth			(f) ButterWorth Estadístico		

Tabla 6.3: Matriz de confusión test con 2º crisis (entrenamiento con 1ª)

en Tabla 6.5 para las validaciones con el entrenamiento de las dos primeras crisis.

Como se puede observar los resultados de los test son muy negativos por lo que se ha desecharido esta línea de investigación

	No Crisis	Crisis		No Crisis	Crisis
No Crisis	86990	0	(a) Datos Brutos		
Crisis	2741	0			
			(b) Datos Estadísticos		
	No Crisis	Crisis		No Crisis	Crisis
No Crisis	86990	0			
Crisis	2741	0			
(c) Filtrado Savitzky–Golay			(d) Savitzky–Golay Estadístico		
	No Crisis	Crisis		No Crisis	Crisis
No Crisis	86990	0			
Crisis	2741	0			
(e) Filtrado ButterWorth			(f) ButterWorth Estadístico		
	No Crisis	Crisis		No Crisis	Crisis
No Crisis	86975	0			
Crisis	2732	0			

Tabla 6.4: Matriz de confusión test 3º crisis (entrenamiento 1º)

		No Crisis	Crisis			No Crisis	Crisis
No Crisis	86713	277		No Crisis	86975	0	
Crisis	2741	0		Crisis	2732	0	
(a) Datos Brutos				(b) Datos Estadísticos			
		No Crisis	Crisis			No Crisis	Crisis
No Crisis	86898	92		No Crisis	86975	0	
Crisis	2741	0		Crisis	2732	0	
(c) Filtrado Savitzky–Golay				(d) Savitzky–Golay Estadístico			
		No Crisis	Crisis			No Crisis	Crisis
No Crisis	86967	23		No Crisis	86975	0	
Crisis	2741	0		Crisis	2732	0	
(e) Filtrado ButterWorth				(f) ButterWorth Estadístico			

Tabla 6.5: Matriz de confusión test 3º crisis (entrenamiento con 1º y 2º)

Capítulo 7

Ensembles para desequilibrados

7.1. Introducción

Otra línea de investigación ha sido la utilización de *ensembles* optimizados para aprendizaje desequilibrado, seguimos dos líneas al utilizar algunos de los algoritmos presentados en la revisión de Galar et. al. [8] y los resultados de Diez et. al. [5]. En particular utilizamos en primera instancia el filtro SMOTE [17] aplicado a *Bagging* y *AdaBoostM1* y *RotationForest* [11].

Los test fueron realizados con *Weka 3.7* y se entrenó con las medias y varianzas móviles de tamaño noventa con las dos primeras crisis, con un SMOTE con un porcentaje de 1000 % y testeado con la tercera crisis con el mismo preprocesado.

7.2. Resultados

Train-Test

Se puede observar en la Tabla 7.1 que siempre tenemos un acierto nulo al testear los datos. Es destacable observar que incluso en el caso de *AdaBoost* (Tabla 7.1c) el entrenamiento falla por lo que podemos deducir que los datos no están bien etiquetados ya que este algoritmo se centra en los datos que falla en predecir y en situaciones de ruido tiene un desempeño peor. Esta prueba además de la etiquetación poco precisa de las crisis existentes nos hace pensar que el sistema realmente es muy ruidoso.

	No Crisis	Crisis		No Crisis	Crisis
No Crisis	203274	0	No Crisis	86936	0
Crisis	1	55274	Crisis	2706	0
(a) Bagging - Train			(b) Bagging - Test		
	No Crisis	Crisis		No Crisis	Crisis
No Crisis	203274	0	No Crisis	86936	0
Crisis	55275	0	Crisis	2706	0
(c) AdaBoostM1 - Train			(d) AdaBoostM1 - Test		
	No Crisis	Crisis		No Crisis	Crisis
No Crisis	203274	0	No Crisis	86936	0
Crisis	1	55274	Crisis	2706	0
(e) Rotation Forest - Train			(f) Rotation Forest - Test		

Tabla 7.1: Matrices de confusión - SMOTE Train-Test

		No Crisis	Crisis			No Crisis	Crisis
No Crisis	203274	0		No Crisis	203274	0	
Crisis		2	5023	Crisis		9	5016
(a) Fold 1-Dos primeras crisis				(b) Fold 2-Dos primeras crisis			
		No Crisis	Crisis			No Crisis	Crisis
No Crisis	290210	0		No Crisis	290209	1	
Crisis		1	7730	Crisis		16	7715
(c) Fold 1 - Todas las crisis				(d) Fold 2 - Todas las crisis			

Tabla 7.2: Matrices de confusión - SMOTE CrossValidation

Validación cruzada

Además de estos experimentos realizamos una validación cruzada con dos subconjuntos de las dos primeras crisis y todas las crisis con el mismo prepocesamiento. Usamos *RotationForest* ya que es el que mejor desempeño demostró en el caso anterior. El desempeño se puede ver en la Tabla 7.2 y aunque el resultado es muy bueno, a la hora de testear el modelo entrenado con las dos primeras crisis con la tercera no se clasifica ningún valor como crisis.

7.3. Conclusiones

De estas pruebas se deducen diferentes conclusiones, primero que los datos que tenemos no están bien etiquetados y son ruidosos y que las diferencias entre las distintas crisis son muy amplias dando a entender que posiblemente no se trata de un sobreajuste sino que los datos entre las distintas crisis está muy separado como pudimos ver en PCA.

Capítulo 8

Random Forest – Media y desviación móviles

8.1. Introducción

En este apartado hemos tratado de encontrar la mejor configuración posible de valores para las ventanas a las que se aplican la media y la desviación de las presiones, con el objetivo de comprobar si es posible entrenar un buen clasificador usando únicamente estas características en lugar de otras más elaboradas y difíciles de calcular. El objetivo era encontrar el mejor área bajo la curva ROC [8], la cual representa el ratio de verdaderos positivos frente a los falsos positivos. Sin embargo, esta métrica puede presentar una visión demasiado optimista del rendimiento del algoritmo cuando se trabaja en el contexto de un conjunto de datos muy desequilibrado [3, 13], como es nuestro caso. Por ello y adicionalmente, realizaremos las mismas operaciones teniendo en consideración una métrica más adecuada para conjuntos desequilibrados, la curva Precision-Recall.

8.2. Modificación del target

Al trabajar con datos calculados a partir de una ventana, hay que tener en cuenta que existen varias formas de considerar como “crisis” una instancia de dato calculada de esta forma. Nosotros definimos 3 formas distintas:

- ‘*Labeled if all are crisis*’: Etiquetamos como “crisis” la instancia si todos los datos de presión que se han usado para calcularla estaban etiquetados como “crisis”.

- ‘Labeled if half are crisis’: Etiquetamos como “crisis” la instancia si al menos la última mitad de los datos de presión que se han usado para calcularla estaban etiquetados como “crisis”.
- ‘Labeled if one is crisis’: Etiquetamos como “crisis” la instancia si el último dato de presión usado para calcularla estaba etiquetado como “crisis”.

8.3. Validación cruzada entre dos días

Para todos los modelos del clasificador Random Forest considerados en los siguientes apartados se empleará el mismo procedimiento de testeo para calcular su rendimiento. Se usarán los datos de 2 días distintos en los que existe una crisis epiléptica y se seguirán los siguientes pasos:

1. Se entrenará el clasificador con los datos del día de la primera crisis y testeado con los de la segunda.
2. Se calculará el rendimiento de acuerdo a la métrica correspondiente (ROC o Precision-Recall según el caso) con los resultados del testeo (auc1¹).
3. Se entrenará el clasificador con los datos del día de la segunda crisis y testeado con los de la primera.
4. Se calculará el rendimiento con los resultados del testeo (auc2²).
5. Se calculará el rendimiento final como la media entre auc1 y auc2. Buscamos que el rendimiento final sea lo mayor posible.

8.4. Primera exploración, métrica=ROC

Hemos realizado una primera exploración combinando de todas las formas posibles las siguientes ventanas y estrategias de modificación del target:

- Ventanas para la media = [5, 30, 55, 80, 105, 130, 155, 180]
- Ventanas para la desviación = [5, 30, 55, 80, 105, 130, 155, 180]

¹AUC del testeo entrenando con la primera crisis

²AUC del testeo entrenando con la segunda crisis

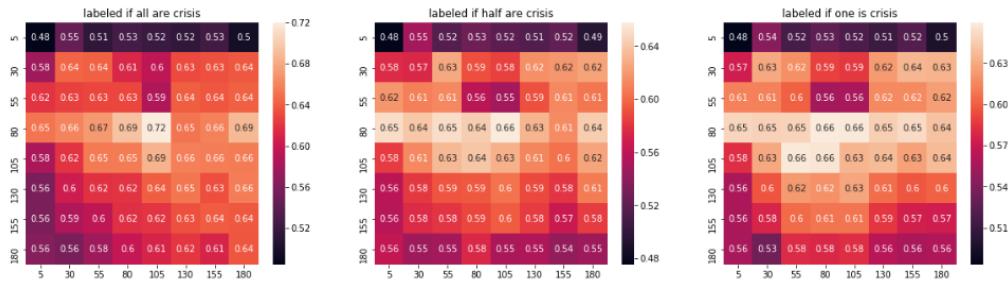


Figura 8.1: Mapa de calor de las áreas bajo la curva calculadas en la primera exploración con la métrica=ROC.

- Modificación del target = ['Labeled if all are crisis', 'Labeled if half are crisis', 'Labeled if one is crisis']

Por cada combinación de ventana para la media, ventana para la desviación, y estrategia de modificación del target se han aplicado ambos cálculos estadísticos a las presiones de 2 días distintos en los que existe una crisis epiléptica y se han usado esos datos estadísticos para llevar a cabo el proceso de validación definido en el apartado anterior.

Tras realizar todas las ejecuciones, la mejor área bajo la curva (ROC = 0.72) se ha encontrado con la siguiente combinación:

- Ventana para la media = 105
- Ventana para la desviación = 80
- Modificación del target = 'Labeled if all are crisis'

En las figuras 8.1, 8.2 y 8.3 el eje vertical representa la ventana para la desviación y el eje horizontal la ventana para la media.

8.5. Segunda exploración, métrica=ROC

A continuación, centramos la exploración en ventanas cercanas a las que han ofrecido mejores resultados en la exploración anterior, teniendo en cuenta que, como se aprecia en la figura Fig. 8.1, el resultado parece ser más dependiente de la ventana de la desviación que la de la media:

- Ventanas para la media = [5, 30, 55, 80, 105, 130, 155, 180]

CAPÍTULO 8. RANDOM FOREST – MEDIA Y DESVIACIÓN MÓVILES

36

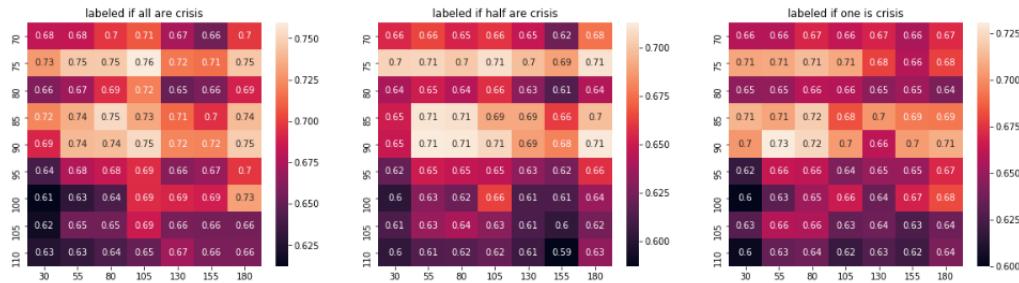


Figura 8.2: Mapa de calor de las áreas bajo la curva calculadas en la segunda exploración con la métrica=ROC.

- Ventanas para la desviación = [70, 75, 80, 85, 90, 95, 100, 105, 110]
- Modificación del target = ['Labeled if all are crisis', 'Labeled if half are crisis', 'Labeled if one is crisis']

Realizando las mismas operaciones, la mejor área bajo la curva ($\text{roc} = 0.76$) se ha encontrado con la siguiente combinación:

- Ventana para la media = 105
- Ventana para la desviación = 75
- Modificación del target = 'Labeled if all are crisis'

8.6. Tercera exploración, métrica=ROC

Finalmente, centramos la exploración un poco más:

- Ventanas para la media = [40, 65, 90, 115, 140, 165, 190]
- Ventanas para la desviación = [80, 82, 84, 86, 88, 90, 92, 94]
- Modificación del target = ['Labeled if all are crisis', 'Labeled if half are crisis', 'Labeled if one is crisis']

La mejor área bajo la curva ($\text{roc}=0.76$) se ha encontrado con la siguiente combinación:

- Ventana para la media = 90

8.7. PRIMERA EXPLORACIÓN, MÉTRICA=PRECISION-RECALL 37

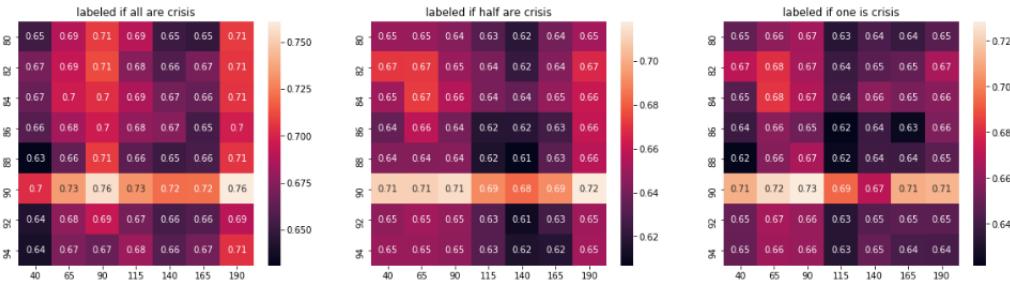


Figura 8.3: Mapa de calor de las áreas bajo la curva calculadas en la tercera exploración con la métrica=ROC.

- Ventana para la desviación = 90
- Modificación del target = ‘Labeled if all are crisis’

8.7. Primera exploración, métrica=Precision-Recall

Al igual que con la otra métrica, hemos realizado una primera exploración combinando de todas las formas posibles las siguientes ventanas y estrategias de modificación del target:

- Ventanas para la media = [5, 30, 55, 80, 105, 130, 155, 180]
- Ventanas para la desviación = [5, 30, 55, 80, 105, 130, 155, 180]
- Modificación del target = [‘Labeled if all are crisis’, ‘Labeled if half are crisis’, ‘Labeled if one is crisis’]

Tras la validación de todas las ejecuciones, la mejor precisión media (average_precision_score = 0.026) se ha encontrado con la siguiente combinación:

- Ventana para la media = 55
- Ventana para la desviación = 105
- Modificación del target = ‘Labeled if one are crisis’

CAPÍTULO 8. RANDOM FOREST – MEDIA Y DESVIACIÓN MÓVILES

38

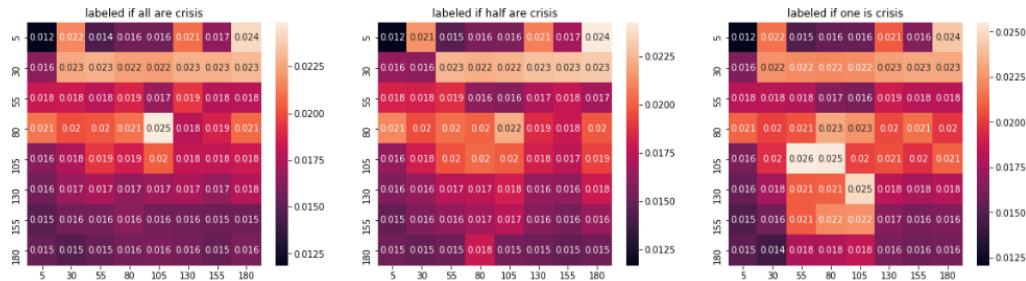


Figura 8.4: Mapa de calor de las áreas bajo la curva calculadas en la primera exploración con la métrica=Precision-Recall.

8.8. Segunda exploración, métrica=Precision-Recall

A continuación, centramos la exploración en ventanas cercanas a las que han ofrecido mejores resultados en la exploración anterior:

- Ventanas para la media = [40, 65, 90, 115, 140, 165, 190]
- Ventanas para la desviación = [70, 75, 80, 85, 90, 95, 100]
- Modificación del target = ['Labeled if all are crisis', 'Labeled if half are crisis', 'Labeled if one is crisis']

Realizando las mismas operaciones, la mejor precisión media (average_precision_score = 0.035) se ha encontrado con la siguiente combinación:

- Ventana para la media = 65
- Ventana para la desviación = 85
- Modificación del target = 'Labeled if one are crisis'

8.9. Tercera exploración, métrica=Precision-Recall

Finalmente, centramos la exploración un poco más:

- Ventanas para la media = [40, 65, 90, 115, 140, 165, 190]

8.9. TERCERA EXPLORACIÓN, MÉTRICA=PRECISION-RECALL 39

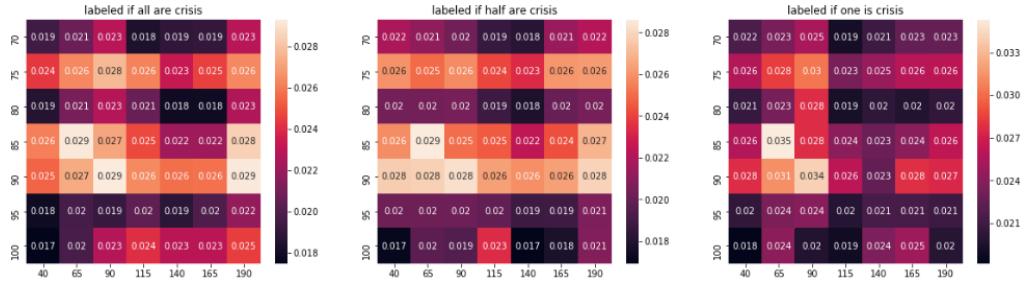


Figura 8.5: Mapa de calor de las áreas bajo la curva calculadas en la segunda exploración con la métrica=Precision-Recall.

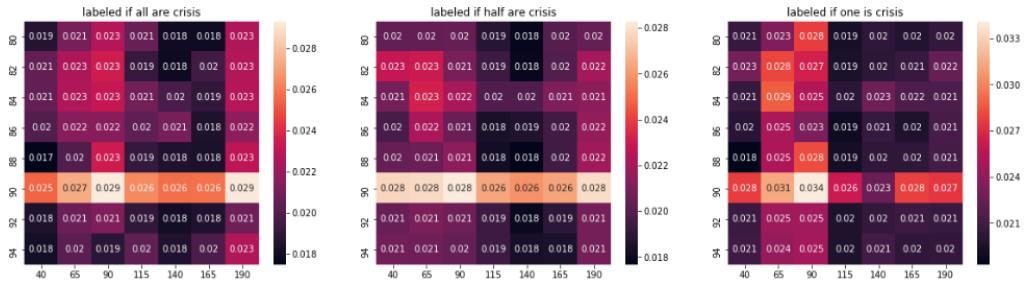


Figura 8.6: Mapa de calor de las áreas bajo la curva calculadas en la tercera exploración con la métrica=Precision-Recall.

- Ventanas para la desviación = [80, 82, 84, 86, 88, 90, 92, 94]
- Modificación del target = ['Labeled if all are crisis', 'Labeled if half are crisis', 'Labeled if one is crisis']

La mejor precisión media (average_precision_score=0.034) se ha encontrado con la siguiente combinación:

- Ventana para la media = 90
- Ventana para la desviación = 90
- Modificación del target = 'Labeled if one are crisis'

Observamos que al igual que con la otra métrica, el valor óptimo de las ventanas resulta ser el mismo, por lo que para cualquier cálculo posterior se empleará un tamaño de ventana de 90 instancias.

Capítulo 9

Extracción de características con *tsfresh* y clasificador Random Forest

9.1. Introducción

Como alternativa a estadísticas más simples como la media y la desviación, hemos utilizado la librería *tsfresh* [2] para extracción de características en series temporales. Dado que en las exploraciones anteriores, obtenemos los mejores resultados con una ventana de 90 tanto para la media como para la desviación, asumimos este valor de ventana para el cálculo de estas características. Utilizando la función `tsfresh.extract_features` con los parámetros por defecto obtenemos 794 características por cada columna. Si lo aplicamos a las columnas relativas a cada uno de los 6 tubos de presión obtenemos un total de 4764 características, demasiadas para entrenar el clasificador, por lo que planteamos varias formas de filtrar las más útiles.

9.2. Filtrado típico de características

A partir de las características obtenidas en el apartado anterior, nuestra primera aproximación es filtrar aplicando una serie de operaciones generales de filtrado una tras otra para ir reduciendo la cantidad de características. Realizamos las siguientes operaciones en este orden:

1. Eliminar las características (columnas) con algún valor nulo, ya que provocará fallos en pasos posteriores.

2. Eliminar las características con formatos no admitidos por los pasos posteriores. En nuestro caso los siguientes filtros fallan al aplicarlos a la característica `__sample_entropy`, por lo que la eliminamos.
3. Eliminar las características con baja varianza usando `sklearn.feature_selection.VarianceThreshold` con un threshold de 0.01.
4. Seleccionar las 1000 mejores características usando `sklearn.feature_selection.SelectKBest` con `score_func=sklearn.feature_selection.chi2` (función chi cuadrado).
5. Seleccionamos las mejores características en función a un modelo. En nuestro caso hemos usado un modelo de clasificación Random Forest con valor de `threshold=0.02` (teniendo en cuenta que los valores están normalizados entre 0 y 1). Este filtrado es supervisado ya que también recibe los valores de target.

Una vez realizados estos pasos nos quedan 7 características. Con estas 7 realizamos la misma operación descrita en el apartado 8.3 (entrenando con los datos de un día y testeando con los de otro, y viceversa) para cada una de las métricas de rendimiento:

- Para la métrica ROC obtenemos un área bajo la curva final de 0.61.
- Para la métrica Precision-Recall obtenemos una precisión media de 0.022.

Ambas mediciones del rendimiento resultan muy lejos de lo deseable. Con distintas variaciones de los parámetros de las operaciones utilizadas obtenemos resultados similares.

9.3. Filtrado mediante la función `select_features` de tsfresh

Dados los malos resultados obtenidos mediante el filtrado del apartado anterior, de forma alternativa hemos probado a usar la función `tsfresh.feature_selection.selection.select_features` para la selección de atributos. Esta selección también es supervisada ya que recibe los targets junto con las características a filtrar. Tras aplicar esta función seguimos teniendo 1731 características, por lo que planteamos una estrategia adicional

9.4. SELECCIÓN DEL MEJOR CONJUNTO DE CARACTERÍSTICAS

para eliminar las menos relevantes. Si asumimos que una característica será relevante si resulta relevante para todos los datos sobre los que se ha calculado, nos quedaremos solo con aquellas características que tras el filtrado, permanezcan para los 6 tubos de presión iniciales. De esta forma nos quedamos únicamente con 744 características (124 por cada tubo de presión).

Aunque el conjunto de características se ha reducido mucho, aún es demasiado grande para aplicarlo a un modelo de clasificación de datos en tiempo real. Esto se debe a que la extracción de características a partir de los datos en bruto es algo lenta, y por esta razón es necesario realizar una selección más acotada.

9.4. Selección del mejor conjunto de características

Tras realizar el paso anterior obtenemos 124 características distintas aplicadas a cada uno de los 6 tubos de presión que podemos emplear para entrenar un clasificador RandomForest. Al mantener las 6 columnas asociadas a cada característica, podemos entrenarlo usando solo una característica (6 columnas de datos) o un conjunto mayor de ellas. Para escoger qué subconjunto de características mínimo nos permite obtener mejores resultados hemos planteado dos estrategias:

- Aplicar el clasificador RandomForest a cada una y realizar un ranking inicial sobre el que trabajar.
- Emplear un algoritmo genético

Ranking en función de la métrica de evaluación

Esta estrategia consiste en aplicar el clasificador RandomForest a cada característica por separado (6 columnas de datos) de la misma forma que en los aparatos anteriores (validación cruzada entre 2 días), y obtener el rendimiento (ya sea mediante ROC o mediante Precision-Recall). Para cada métrica generamos un ranking, el cual sitúa más arriba a aquellas características que hayan obtenido un rendimiento mayor.

- La mejor característica usando la ROC como métrica (`__change_-quantiles(qh=1.0,ql=0.4)`) obtiene un área bajo la curva de 0.8.

CAPÍTULO 9. EXTRACCIÓN DE CARACTERÍSTICAS CON TSFRESH Y CLASIFICADOR RANDOM FOREST

44

- La mejor característica usando Precision-Recall como métrica (`__change_quantiles(qh=1.0, ql=0.4)`) obtiene un rendimiento medio de 0.099.

A partir de cada uno de los rankings podemos hacer combinaciones de varias formas:

- En primer lugar probamos a ir añadiendo características en el orden en el que se sitúan en el ranking para el entrenamiento del clasificador. Probaremos con las características 1 y 2, las 1, 2 y 3, las 1, 2, 3 y 4... y así sucesivamente hasta que el área bajo la curva obtenida deje de aumentar.
 - Usando la ROC como métrica comprobamos que la combinación de las características 1 y 2 ya ofrece un área bajo la curva peor, de 0.78, por lo que parece probable que combinar características buenas no produce necesariamente mejores resultados.
 - Usando Precision-Recall como métrica obtenemos que la combinación de las características 1 y 2, (`__change_quantiles(f_agg="mean", isabs=True, qh=1.0, ql=0.4)`) y `__number_peaks(n=1)` ofrece un rendimiento medio algo mejor, de 0.107. Sin embargo la combinación 1, 2 y 3 ya ofrece peores resultados.
- En segundo lugar probamos a combinar la mejor característica con todas las demás y comprobar si el área bajo la curva aumenta. Probadmos así con las características 1 y 2, las 1 y 3, las 1 y 4... y así sucesivamente.
 - Usando la ROC, en este caso comprobamos que el área bajo la curva mejora con dos de las combinaciones. Con las combinaciones de las características 1 y 3 (`__change_quantiles(f_agg="mean", isabs=True, qh=1.0, ql=0.4)`) y `__change_quantiles(f_agg="var", isabs=False, qh=1.0, ql=0.4)`), y las características 1 y 5 (`__change_quantiles(f_agg="mean", isabs=True, qh=1.0, ql=0.4)`) y `__symmetry_looking(r=0.25)`) se obtiene un área bajo la curva de 0.83.
 - Usando Precision-Recall encontramos 32 combinaciones que mejoran la precisión media, siendo la combinación de las características 1 y 11 (`__change_quantiles(f_agg="mean", isabs=True,`

9.4. SELECCIÓN DEL MEJOR CONJUNTO DE CARACTERÍSTICAS

`qh=1.0, ql=0.4)` y `__change_quantiles(f_agg="var", isabs=True, qh=0.8, ql=0.4)`) la que ofrece un mejor resultado, de 0.142.

Algoritmo genético

Como alternativa a los métodos planteados en el apartado anterior probamos un algoritmo genético para la selección de la mejor combinación de características. Para ello vamos a usar la librería deap [7], un framework de python para computación evolutiva.

- **Genotipo:** A partir del ranking de características calculado en el apartado anterior, planteamos un genotipo en el que cada individuo puede incluir una combinación de como máximo 10 características. Para ello usamos un array de tamaño 10 que contiene números enteros. Cada gen (número entero) identifica una de las características del ranking (realizaremos una ejecución para cada uno de los 2 rankings, es decir, para cada una de las métricas planteadas).
- **Fenotipo:** La evaluación de cada individuo consistirá en entrenar y testear mediante validación cruzada de 2 días el clasificador Random Forest, usando las características indicadas por los genes del individuo concreto. El valor de fitness del individuo corresponderá con el valor final del rendimiento (calculado dependiendo de la métrica escogida) calculado por este método. Dado que utilizamos un método de cruce no adecuado para permutaciones, es posible que en el genotipo aparezcan genes repetidos. En este caso a la hora de evaluar solo se añadirá esa característica una vez, haciendo que el conjunto final de características pueda ser menor de 10.
- **Método de cruce:** Cruce uniforme con probabilidad de 0.5 por cada gen.
- **Método de mutación:** Mutación por modificación de gen por otro permitido con una probabilidad de 0.2 por gen.
- **Método de selección:** Selección por torneo con tamaño 3.
- **Probabilidad de cruce:** Se aplicará la operación de cruce sobre un individuo de la población con una probabilidad de 0.6.
- **Probabilidad de mutación:** Se aplicará la operación de mutación sobre un individuo de la población con una probabilidad de 0.1.

- **Tamaño de la población:** 50 individuos.
- **Número de generaciones:** 50. El número de generaciones se ha escogido teniendo en cuenta lo costoso que es el procedimiento de evaluación de cada individuo, pero consideramos que ha resultado ser suficiente ya que no se ha producido ninguna mejora en las últimas 18 generaciones.

Resultados usando la ROC como métrica:

Tras la evolución el mayor valor de área bajo la curva que se ha logrado es de 0.8587 con las características indicadas por el individuo [28, 54, 23, 68, 83, 97, 61, 120, 44, 64]. Es el mayor valor de área bajo la curva conseguido hasta el momento. Las características indicadas por el mejor individuo son las siguientes:

- **longest_strike_below_mean:** La longitud de la subsecuencia consecutiva más larga que es mayor que la media.
- **symmetry_looking:** Variable booleana que denota si la distribución es simétrica, con parámetro $r=0.6$.
- **change_quantiles:** Calcula el promedio del valor absoluto de los cambios consecutivos de la serie entre los cuantiles 0.8 y 0.4. $f_agg="mean"$, $isabs=True$.
- **symmetry_looking:** Variable booleana que denota si la distribución es simétrica, con parámetro $r=0.2$.
- **quantile:** Calcula el cuantil 0.8.
- **large_standard_deviation:** Variable booleana que denota si la desviación estándar es mayor que 0.55 veces el rango.
- **large_standard_deviation:** Variable booleana que denota si la desviación estándar es mayor que 0.35 veces el rango.
- **cwt_coefficients:** Calcula una transformada de ondícula continua para la ondícula de Ricker.
- **number_peaks:** Calcula el número de picos de anchura 5.
- **symmetry_looking:** Variable booleana que denota si la distribución es simétrica, con parámetro $r=0.25$.

9.4. SELECCIÓN DEL MEJOR CONJUNTO DE CARACTERÍSTICAS

A primera vista y sin ahondar mucho en el significado de cada característica, podemos observar que existe redundancia en el individuo, ya que calcula tres veces la característica (`symmetry_looking`) y dos veces (`large_standard_deviation`) aunque con parámetros distintos.

En la Figura 9.1 se aprecia la evolución de cada población del algoritmo genético. La primera gráfica (verde) representa el área bajo la curva del mejor individuo de cada generación, mientras que la segunda (roja) representa la del peor individuo. En la primera podemos apreciar que en las últimas generaciones la mejor área bajo la curva se estanca en un valor próximo a 0.86. Apreciamos que la gráfica verde no es estrictamente creciente ya que la función de evolución empleada no implementa elitismo. Sin embargo, sí guarda un histórico de los mejores individuos aunque estos no se inyecten en la siguiente generación, y será de ahí de donde se saque el mejor individuo aunque este no se encuentre en la última población. Las gráficas azul y amarilla representan, respectivamente, la media y la desviación de las áreas bajo la curva de cada generación. Al conseguir mejores individuos, la media también aumenta con cada generación, pero como se observa en la última gráfica (amarilla), es conveniente que la desviación no disminuya demasiado, ya que peores individuos, generados tanto por mutación como por cruce, son necesarios para mantener la diversidad que permite la mejora de los mejores individuos.

Resultados usando Precision-Recall como métrica:

Tras la evolución la mayor precisión media que se ha logrado es de 0.1990 con las características indicadas por el individuo [24, 54, 39, 3, 17, 21, 74, 36, 73, 5]. Las características indicadas por el mejor individuo son las siguientes:

- `agg_linear_trend` Calcula una regresión lineal de mínimos cuadrados para los valores de las series de tiempo que se agregaron a lo largo de los fragmentos de tamaño 5 en comparación con la secuencia desde 0 hasta el número de fragmentos menos uno. `f_agg="var",attr=intercept`.
- `symmetry_looking` Variable booleana que denota si la distribución es simétrica, con parámetro `r=0.6`.
- `change_quantiles` Calcula el promedio del valor absoluto de los cambios consecutivos de la serie entre los cuantiles 0.1 y 0.2. `f_agg="var",isabs=False`.
- `change_quantiles` Entre los cuantiles 0.8 y 0.4. `f_agg="var",isabs=False`.

**CAPÍTULO 9. EXTRACCIÓN DE CARACTERÍSTICAS CON
TSFRESH Y CLASIFICADOR RANDOM FOREST**

48

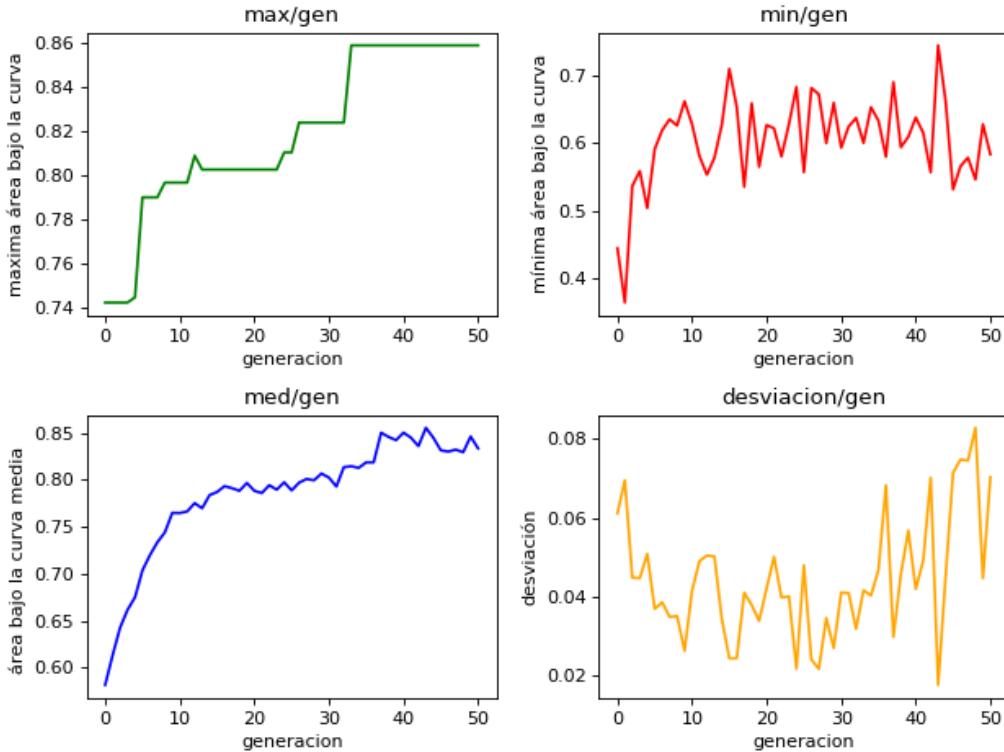


Figura 9.1: Gráficas de la evolución de las poblaciones en el algoritmo genético con la métrica=ROC.

- `change_quantiles` Entre los cuantiles 0.6 y 0.4. `f_agg="var"`,`isabs=True`.
- `last_location_of_minimum` Devuelve la última ubicación del valor mínimo de la serie.
- `number_peaks` Calcula el número de picos de al menos anchura 1 en la serie.
- `change_quantiles` Entre los cuantiles 0.1 y 0.4. `f_agg="mean"`,`isabs=True`.
- `change_quantiles` Entre los cuantiles 1.0 y 0.4. `f_agg="mean"`,`isabs=True`.
- `agg_linear_trend` Con fragmentos de tamaño 5. `f_agg="min"`,`attr="stderr"`.

En este caso también aparece el mismo atributo (`change_quantiles` y `agg_linear_trend`) con distintos parámetros varias veces en el genotipo.

Por otro lado en la Figura 9.2 se aprecia que la primera gráfica (verde), que representa la precisión media del mejor individuo de cada generación, no

9.4. SELECCIÓN DEL MEJOR CONJUNTO DE CARACTERÍSTICAS

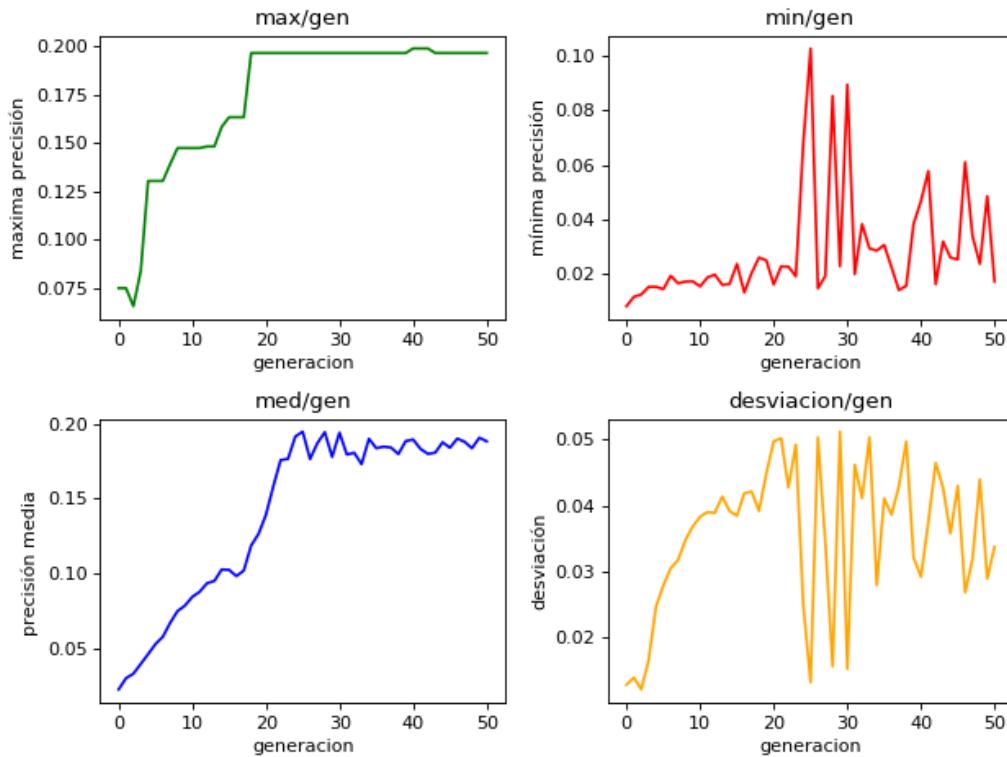


Figura 9.2: Gráficas de la evolución de las poblaciones en el algoritmo genético con la métrica=Precision-Recall.

es estrictamente creciente. Esto puede ocurrir al perder el mejor individuo de la generación al ser mutado o cruzado, sin embargo Deap lleva un histórico de los mejores individuos de cada generación, y al terminar la evolución devuelve el mayor de ellos, no el mejor de la última generación, por lo que el genotipo presentado sí corresponde con el mejor individuo encontrado por el algoritmo genético.

Capítulo 10

Ensembles para desequilibrados

tsfresh

10.1. Introducción

Tras obtener el conjunto de datos que optimiza la curva *Precision-Recall* se han probado diversos *ensembles* para datos desequilibrados. Se han probado tres métodos de remuestreo: *Random Balance* [4], *SMOTE* [8] y *Random under sampling* [5]. Se han usado justo a estos los métodos de *Bagging* [8], *Rotation Forest* [11] y *RandomCommittee* [5]. Se ha desecharido el uso de métodos de *boosting* ya que los resultados de los experimentos del capítulo 7 pudimos observar que los datos mal etiquetados afectaban mucho al entrenamiento.

Se ha realizado un entrenamiento con una crisis y testeado con la otra. Se han realizado experimentos utilizando la herramienta Weka [9]. Cada ejemplo se ha ejecutado 10 veces y se ha realizado la media de los resultados. Estos a su vez se han normalizado entre 0 y 1 y son:

- **TPR:** *True positive rate*
- **FPR:** *False positive rate*
- **TNR:** *True negative rate*
- **FNR:** *False negative rate*
- **PRC:** *Precision-Recall Curve*
- **AUC:** *Area under the ROC Curve*

	TPR	FPR	TNR	FNR	PRC	AUC	ACC
RB - Bag	0	0	1	1	0.984425	0.552363	0.98178
RB - RC	0	0	1	1	0.981267	0.485675	0.98178
RB - RotF	0	1.65036e-06	0.999998	1	0.980551	0.485354	0.981778
RUS - Bag	0	0.000899444	0.999101	1	0.980613	0.447139	0.980897
RUS - RC	0	0	1	1	0.981076	0.4806	0.98178
RUS - RotF	0	0.000394435	0.999606	1	0.982287	0.486412	0.981393
SM - Bag	0	0.000173287	0.999827	1	0.984367	0.565807	0.98161
SM - RC	0	0	1	1	0.981278	0.486071	0.98178
SM - RotF	0	1.56784e-05	0.999984	1	0.982673	0.512364	0.981764

Tabla 10.1: Métodos para desbalanceados - Entrenamiento con la primera crisis, testeo con la segunda crisis.

- **ACC:** *Accuracy*

Las abreviaturas para los métodos son:

- **RB:** *Random Balance*
- **RUS:** *Random Undersampling*
- **SM:** *SMOTE*
- **Bag:** *Bagging*
- **RC:** *Random Committee*
- **RotF:** *Rotation Forest*

10.2. Resultado de experimentos

Los resultados de entrenar con la primera crisis y testear con la segunda se puede ver en la tabla 10.1 y en la figura 10.1. El resultado de la misma operación pero entrenando con la segunda crisis y testeando con la primera está en la tabla 10.2 y en la figura 10.2.

10.3. Comentario de los resultados

Como se puede observar gracias a buscar un conjunto de características que optimizan el valor de PRC este valor en todos los experimentos es muy alto. Sin embargo, en todos los experimentos nunca es capaz ningún modelo

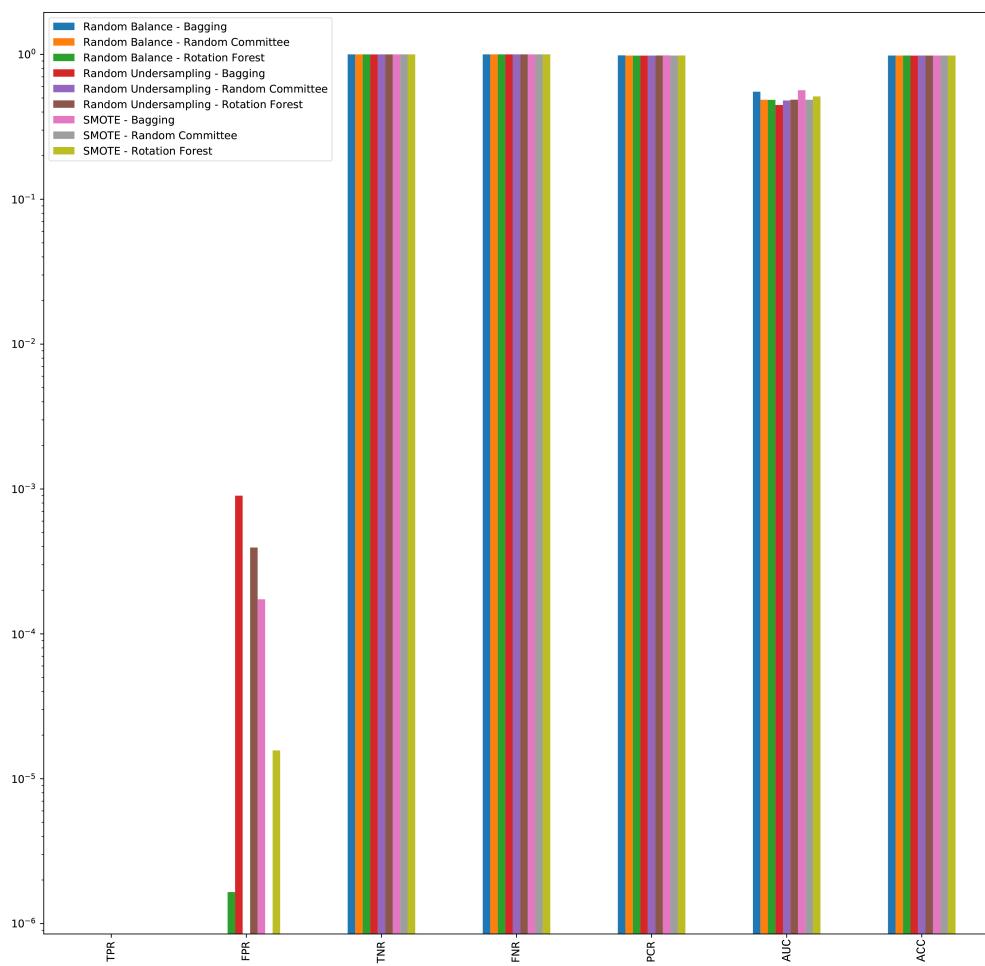


Figura 10.1: Métodos para desbalanceados - Entrenamiento con la primera crisis, testeо con la segunda crisis.

54 CAPÍTULO 10. ENSEMBLES PARA DESEQUILIBRADOS TSFRESH

	TPR	FPR	TNR	FNR	PCR	AUC	ACC
RB - Bag	0	0.0109634	0.989037	1	0.995225	0.491511	0.983616
RB - RC	0	0.00571683	0.994283	1	0.993912	0.446135	0.988834
RB - RotF	0	0.00488364	0.995116	1	0.99517	0.525534	0.989663
RUS - Bag	0	0.0461306	0.953869	1	0.994863	0.514533	0.948642
RUS - RC	0	0.0112769	0.988723	1	0.994602	0.506568	0.983304
RUS - RotF	0	0.0128567	0.987143	1	0.994482	0.468075	0.981733
SM - Bag	0	0.017596	0.982404	1	0.995505	0.539099	0.97702
SM - RC	0	0.0048424	0.995158	1	0.994052	0.458383	0.989704
SM - RotF	0	0.00344412	0.996556	1	0.994742	0.502318	0.991094

Tabla 10.2: Métodos para desbalanceados - Entrenamiento con la segunda crisis, testeo con la primera crisis.

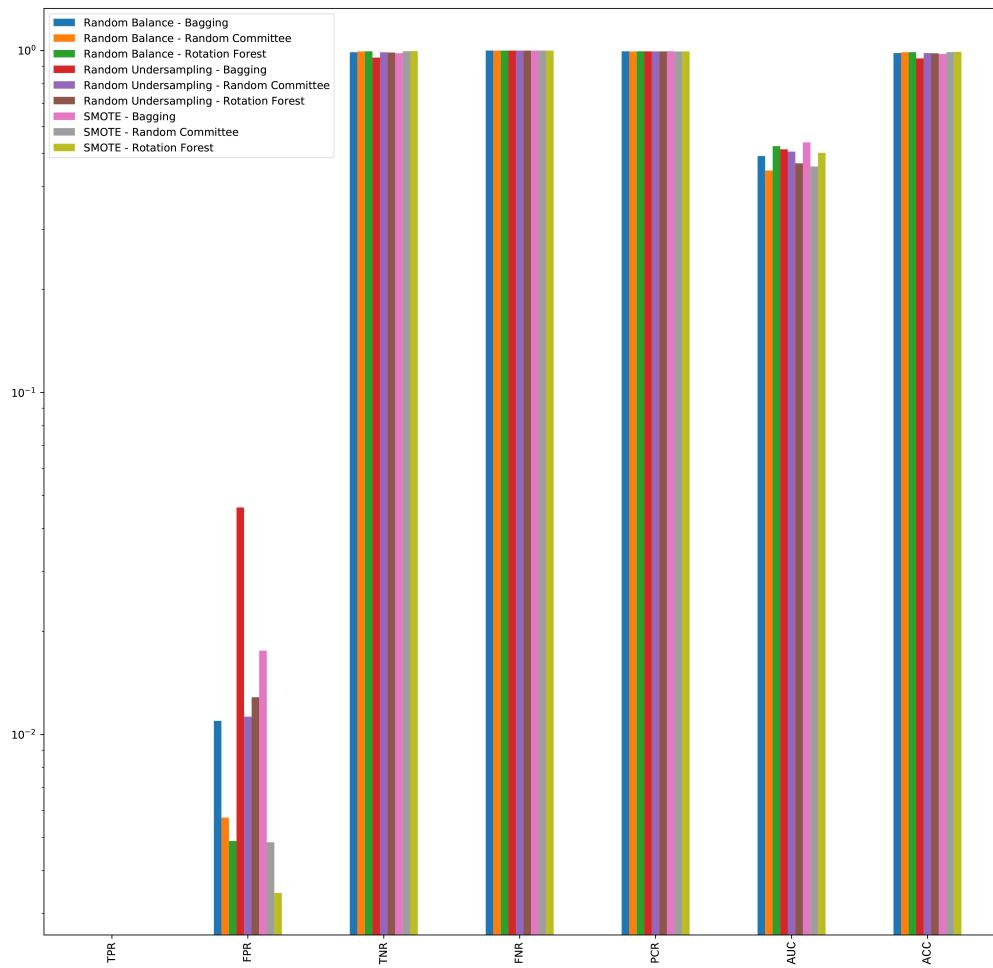


Figura 10.2: Métodos para desbalanceados - Entrenamiento con la segunda crisis, testeo con la primera crisis.

de predecir correctamente una situación de crisis y las únicas veces que se ha obtenido como resultado de la predicción una situación de crisis han sido erróneas.

A destacar que los métodos que menos error han tenido han sido los que usan *Random Balance* y *SMOTE* y los algoritmos *Rotation Forest* y *Random Committee*.

10.4. Conclusiones

Tras realizar toda esta la investigación e intentar obtener los mejores resultados probando la mayor cantidad técnicas que hemos podido explorar, lamentablemente, debido a la limitación de los datos de crisis y los problemas de como están etiquetados los datos no se ha podido encontrar un modelo que pueda ser usado en producción ya que ningún modelo probado ha sido capaz de predecir correctamente situaciones de crisis. Sin embargo, se espera que si en algún momento se obtuvieran datos suficientes y de calidad, estos mismos experimentos se podrían emplear para encontrar un modelo efectivo.

Bibliografía

- [1] Ingwer Borg and Patrick Groenen. Modern multidimensional scaling: Theory and applications. *Journal of Educational Measurement*, 40(3):277–280, 2003.
- [2] Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W Kempa-Liehr. Time series feature extraction on basis of scalable hypothesis tests (tsfresh—a python package). *Neurocomputing*, 307:72–77, 2018.
- [3] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML ’06, pages 233–240, New York, NY, USA, 2006. ACM.
- [4] José F Díez-Pastor, Juan J Rodríguez, César García-Osorio, and Ludmila I Kuncheva. Random balance: ensembles of variable priors classifiers for imbalanced data. *Knowledge-Based Systems*, 85:96–111, 2015.
- [5] José F Díez-Pastor, Juan J Rodríguez, César I García-Osorio, and Ludmila I Kuncheva. Diversity techniques improve the performance of the best imbalance learning ensembles. *Information Sciences*, 325:98–117, 2015.
- [6] Clinton Dreisbach. Building scikit-learn transformers. <https://dreisbach.us/articles/building-scikit-learn-compatible-transformers/>, Jun 2015.
- [7] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175, jul 2012.

- [8] Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484, 2012.
- [9] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.
- [10] Joseph B Kruskal. Nonmetric multidimensional scaling: a numerical method. *Psychometrika*, 29(2):115–129, 1964.
- [11] Juan José Rodriguez, Ludmila I Kuncheva, and Carlos J Alonso. Rotation forest: A new classifier ensemble method. *IEEE transactions on pattern analysis and machine intelligence*, 28(10):1619–1630, 2006.
- [12] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.
- [13] Takaya Saito and Marc Rehmsmeier. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PLOS ONE*, 10(3):1–21, 03 2015.
- [14] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.
- [15] Wikipedia contributors. Radial basis function — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Radial_basis_function&oldid=875832599, 2018. [Online; accessed 18-February-2019].
- [16] Wikipedia contributors. Butterworth filter — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Butterworth_filter&oldid=881318917, 2019. [Online; accessed 18-February-2019].
- [17] Wikipedia contributors. Oversampling and undersampling in data analysis — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Oversampling_and_undersampling_in_data_analysis&oldid=887800096, 2019. [Online; accessed 28-March-2019].

- [18] Wikipedia contributors. Savitzky–golay filter — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Savitzky%20%93Golay_filter&oldid=877315967, 2019. [Online; accessed 18-February-2019].
- [19] Zhenyue Zhang and Jing Wang. Mlle: Modified locally linear embedding using multiple weights. In *Advances in neural information processing systems*, pages 1593–1600, 2007.