



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
detección de crisis epilépticas
y aplicación Android**

Documentación Técnica



Presentado por Alicia Olivares Gil
en Universidad de Burgos — 1 de julio
de 2019

Tutores: Álvar Arnaiz González y José
Franciso 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	13
Apéndice B Especificación de Requisitos	17
B.1. Introducción	17
B.2. Objetivos generales	17
B.3. Catalogo de requisitos	17
B.4. Especificación de requisitos	19
Apéndice C Especificación de diseño	33
C.1. Introducción	33
C.2. Diseño de datos	33
C.3. Diseño procedimental	34
C.4. Diseño arquitectónico	36
C.5. Diseño de interfaces	38
Apéndice D Documentación técnica de programación	41
D.1. Introducción	41
D.2. Estructura de directorios	41

D.3. Manual del programador	43
D.4. Compilación, instalación y ejecución del proyecto	44
D.5. Pruebas del sistema	47
Apéndice E Documentación de usuario	49
E.1. Introducción	49
E.2. Requisitos de usuarios	49
E.3. Instalación	50
E.4. Manual del usuario	50
Bibliografía	59

Índice de figuras

B.1. Diagrama de casos de uso, nivel 0	20
B.2. Diagrama de casos de uso, visualización de datos.	20
B.3. Diagrama de casos de uso, administración de usuarios.	21
B.4. Diagrama de casos de uso, administración de camas.	21
C.1. Diagrama de secuencias, llamada a los servicios genéricos de la API.	35
C.2. Diagrama de secuencias, petición de datos en tiempo real.	36
C.3. Abstracción de la arquitectura de microservicios.	37
C.4. Prototipos iniciales de las pantallas de: login, administración, visualización de camas y visualización de datos.	38
C.5. Paleta de colores.	39
C.6. Interfaces de usuario finales de las pantallas de: login, administración, visualización de camas y visualización de datos.	40
D.1. Estructura de directorios del repositorio.	43
D.2. Descarga del contenido del repositorio.	45
D.3. Importar proyecto de Android Studio.	45
E.1. Iniciar sesión.	51
E.2. Menú de administración.	52
E.3. Gestión de usuarios, cambiar contraseña.	53
E.4. Gestión de usuarios, eliminar y añadir usuarios.	53
E.5. Gestión de camas, asignar usuarios.	54
E.6. Visualización de camas.	55
E.7. Visualización de los datos de una cama en tiempo real.	56
E.8. Menú de navegación para un administrador y para un usuario genérico.	57

E.9. Modificar contraseña.	57
E.10. Acerca de la aplicación.	58

Índice de tablas

A.1.	Tareas del sprint 1	2
A.2.	Tareas del sprint 2	2
A.3.	Tareas del sprint 3	3
A.4.	Tareas del sprint 4	3
A.5.	Tareas del sprint 5	3
A.6.	Tareas del sprint 6	4
A.7.	Tareas del sprint 7	4
A.8.	Tareas del sprint 8	5
A.9.	Tareas del sprint 9	5
A.10.	Tareas del sprint 10	6
A.11.	Tareas del sprint 11	6
A.12.	Tareas del sprint 12	6
A.13.	Tareas del sprint 13	7
A.14.	Tareas del sprint 14	7
A.15.	Tareas del sprint 15	8
A.16.	Tareas del sprint 16	8
A.17.	Tareas del sprint 17	9
A.18.	Tareas del sprint 18	9
A.19.	Tareas del sprint 19	9
A.20.	Tareas del sprint 20	10
A.21.	Tareas del sprint 21	10
A.22.	Tareas del sprint 22	11
A.23.	Tareas del sprint 23	11
A.24.	Tareas del sprint 24	11
A.25.	Tareas del sprint 25	12
A.26.	Tareas del sprint 26	12
A.27.	Coste de cada sprint	13

A.28.Costes de personal	14
A.29.Costes de <i>hardware</i>	14
A.30.Coste total.	15
B.1. Caso de uso 1: Iniciar sesión	22
B.2. Caso de uso 2: Visualizar de datos	23
B.3. Caso de uso 2.1: Elegir cama	23
B.4. Caso de uso 2.2: Ver datos en tiempo real	24
B.5. Caso de uso 3: Administrar de usuarios	25
B.6. Caso de uso 3.1: Añadir usuarios	26
B.7. Caso de uso 3.2: Modificar contraseña	27
B.8. Caso de uso 3.3: Borrar usuario	28
B.9. Caso de uso 4: Administrar de camas	28
B.10.Caso de uso 4.1: Añadir cama	29
B.11.Caso de uso 4.2: Modificar cama	30
B.12.Caso de uso 4.3: Borrar cama	30
B.13.Caso de uso 4.4: Asignar cama a usuario	31
D.1. Pruebas unitarias.	47
D.2. Pruebas de interfaz.	48

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En este apartado se va exponer la planificación temporal del proyecto, indicando qué tareas y cuándo se realizaron. Además, se presenta un análisis de la viabilidad legal y económica del proyecto.

A.2. Planificación temporal

La planificación temporal se ha realizado adaptando la metodología *Scrum* a un proyecto educativo, con los cambios que esto conlleva.

- El desarrollo se ha basado en iteraciones o *sprints* de una semana de duración aproximadamente.
- Cada uno de los *sprints* contiene las tareas o *issues* que se realizaron esa semana.
- Cada tarea tiene asociado un coste, que simboliza su dificultad en cuanto al esfuerzo que se estima invertir en ella.
- En caso de que la estimación del coste resultara inexacta al realizar el *issue*, este se modificó para reflejar el esfuerzo real empleado.
- Al finalizar cada *sprint* se realizaba una reunión de revisión con los tutores donde se exponían los progresos realizados y se planificaba el siguiente *sprint*.

Sprint 1

Fecha: 19/12/2018 - 23/12/2018

El primer *sprint* consistió en realizar una exploración bibliográfica inicial sobre el estado del arte.

<i>Issue</i>	Estimado	Final
Crear y configurar repositorio	2	2
Exploración bibliográfica inicial	13	13

Tabla A.1: Tareas del sprint 1

Sprint 2

Fecha: 23/12/2018 - 29/12/2018

Se continuó la exploración bibliográfica inicial, centrándose en artículos especialmente interesantes encontrados hasta el momento y se comenzó la exploración bibliográfica sobre otros métodos aplicables al problema.

<i>Issue</i>	Estimado	Final
Continuación de la exploración bibliográfica inicial	8	8
Exploración bibliográfica sobre otros métodos aplicables al problema	8	8
Lectura de «Automated Epileptic Seizure Detection Methods: A Review Study»	8	8
Instalar y configurar cliente VPN	2	2

Tabla A.2: Tareas del sprint 2

Sprint 3

Fecha: 29/12/2018 - 11/01/2019

Se inició la documentación y se empezó a trabajar en la visualización de los datos en bruto y de algunos datos estadísticos.

Issue	Estimado	Final
Iniciar documentación de los sprints	-	5
Instalar entorno y librerías de Python	5	5
Aprender a usar librerías	8	8
Procesar y mostrar datos	8	8

Tabla A.3: Tareas del sprint 3

Sprint 4

Fecha: 11/01/2019 - 18/01/2019

Se configuró el acceso al equipo de cómputo del grupo de investigación para probar técnicas de reducción de la dimensionalidad de los datos y algunas opciones básicas de filtrado y suavizado de la señal.

Issue	Estimado	Final
Configurar acceso a gamma	5	5
Probar opciones filtrado y suavizado	8	8
Probar otras formas de proyección de datos	8	21

Tabla A.4: Tareas del sprint 4

Sprint 5

Fecha: 18/01/2019 - 25/01/2019

Se hicieron cambios en el preprocesado, se probaron otras formas de filtrado de la señal y se estudiaron los puntos clave de las proyecciones del *sprint* anterior.

Issue	Estimado	Final
Leer apuntes de minería de datos	8	8
Modificar preprocesado	3	3
Representar señales en torno a la crisis epiléptica	5	5
Probar formas de filtrado de la señal	5	5
Estudiar los puntos clave de las proyecciones	5	8

Tabla A.5: Tareas del sprint 5

Sprint 6

Fecha: 25/01/2019 - 31/01/2019

Se centraron las pruebas en las proyecciones con mejor rendimiento, concretamente en MDS [9], y se iniciaron la documentación de la planificación temporal y el cuaderno de investigación.

<i>Issue</i>	Estimado	Final
Cambiar a proyecciones con mejor rendimiento	13	13
Pasar cálculos estadísticos a funciones	5	5
Documentar 5 primeros Sprints en el Plan de Proyecto	8	8
Documentar investigación en Overleaf [11]	5	5

Tabla A.6: Tareas del sprint 6

Sprint 7

Fecha: 31/01/2019 - 07/02/2019

Se codificaron las transformaciones generadas en los *sprints* anteriores (normalización, filtros y estadísticas) como transformadores de Sklearn [12].

<i>Issue</i>	Estimado	Final
Aprender sobre la clase TransformerMixin del paquete sklearn.base [7]	3	3
Generar transformadores para las funciones usadas	21	13

Tabla A.7: Tareas del sprint 7

Sprint 8

Fecha: 07/02/2019 - 14/02/2019

Se exploraron otras formas de proyección y se realizó una primera aproximación de clasificación mediante Random Forest [8] y detección de anomalías One-class [10].

<i>Issue</i>	Estimado	Final
Probar Kernel PCA	5	5
Acotar ataque a partir del aspecto de la señal y la salida de las proyecciones (MDS)	3	5
Probar MDS con el ataque reetiquetado	8	8
Probar clasificador Random Forest	8	8
Aplicar detección de anomalías one-class	13	13

Tabla A.8: Tareas del sprint 8

Sprint 9

Fecha: 14/02/2019 - 21/02/2019

Se planteó la evaluación de los clasificadores mediante el área bajo la curva ROC y se terminaron de documentar las proyecciones en el cuaderno de investigación.

<i>Issue</i>	Estimado	Final
Valorar los resultados de Random Forest mediante el área bajo la curva	3	3
Incluir las proyecciones en la documentación	5	5
Preparar la visualización de las proyecciones para la documentación	5	8

Tabla A.9: Tareas del sprint 9

Sprint 10

Fecha: 21/02/2019 - 28/02/2019

Una parte se invirtió en aprender sobre clasificación de conjuntos de datos desequilibrados mediante ensambles y por otro lado se realizó una exploración de ventanas para la aplicación de los datos al clasificador Random Forest.

<i>Issue</i>	Estimado	Final
Lectura de aprendizaje sobre datos desequilibrados	-	8
Aplicar Random Forest a datos estadísticos con distintas ventanas	8	8

Tabla A.10: Tareas del sprint 10

Sprint 11

Fecha: 28/02/2019 - 07/03/2019

Se continuó con la lectura sobre desequilibrados y se inició el aprendizaje sobre la librería tsfresh [2] para extracción de características en series temporales.

<i>Issue</i>	Estimado	Final
Continuar con la lectura sobre uso de ensembles para conjuntos desequilibrados	5	5
Extracción de características en series temporales	8	8

Tabla A.11: Tareas del sprint 11

Sprint 12

Fecha: 07/03/2019 - 14/03/2019

Se trataron de aplicar los resultados de la extracción de características de series temporales al clasificador Random Forest.

<i>Issue</i>	Estimado	Final
Random Forest con características de series temporales	5	5
Continuar extracción de características en series temporales	13	13

Tabla A.12: Tareas del sprint 12

Sprint 13

Fecha: 14/03/2019 - 21/03/2019

Principalmente se exploraron formas de filtrar y combinar las mejores características de series temporales para ser aplicadas al clasificador.

Issue	Estimado	Final
Filtrado de características	13	5
Aplicar Random Forest a combinaciones de las mejores características	5	5
Documentación de sprints pasados y actualización del cuaderno de investigación	5	5

Tabla A.13: Tareas del sprint 13

Sprint 14

Fecha: 21/03/2019 - 28/03/2019

Se planteó un filtrado de características mediante un algoritmo genético usando el framework DEAP [3] de python y se realizó una investigación inicial de técnicas para la implementación de servidores de *streaming*.

Issue	Estimado	Final
Investigar técnicas para implementar servidores de streaming	8	8
Algoritmo genético para la selección de características	13	13
Avanzar con la documentación en el cuaderno de investigación	5	13

Tabla A.14: Tareas del sprint 14

Sprint 15

Fecha: 28/03/2019 - 04/04/2019

Se mejoró el algoritmo genético, se finalizó su ejecución con la ayuda de tmux [13] y se documentaron los resultados. Además, se inició el diseño de

los requisitos y los casos de uso de la aplicación y se plantearon los primeros prototipos.

<i>Issue</i>	Estimado	Final
Generar prototipo de la pantalla de visualización de datos	5	5
Aprender sobre tmux para la ejecución del genético	3	3
Mejorar algoritmo genético y documentar resultados	8	8
Plantear primeras cuestiones de diseño de la app	5	5

Tabla A.15: Tareas del sprint 15

Sprint 16

Fecha: 04/04/2019 - 11/04/2019

Se ultimaron los detalles del cuaderno de investigación con la documentación generada hasta el momento, se finalizaron los prototipos y se documentó la parte de diseño y de las técnicas. Además, se instaló Android Studio para su uso en sprints posteriores.

<i>Issue</i>	Estimado	Final
Ultimar detalles del cuaderno e trabajo	-	8
Finalizar y documentar prototipos	5	5
Avanzar en la documentación temporal, de diseño y de las técnicas	13	13
Instalar Android Studio	2	2

Tabla A.16: Tareas del sprint 16

Sprint 17

Fecha: 11/04/2019 - 18/04/2019

Se refactorizó el código de los experimentos para incluir el testeo mediante la métrica precision-recall, más adecuada para conjuntos de datos desequilibrados, y se volvieron a ejecutar los filtrados de características.

A.2. PLANIFICACIÓN TEMPORAL	9
-----------------------------	---

Issue	Estimado	Final
Refactorizar el código de Random Forest para incluir la métrica precision-recall	8	8
Volver a ejecutar los filtrados de características para la nueva métrica	8	8

Tabla A.17: Tareas del sprint 17

Sprint 18

Fecha: 18/04/2019 - 02/05/2019

Se documentaron los resultados de los filtrados de características del sprint anterior y se comenzó la lectura sobre la documentación de Android Studio y la visualización del curso *Android Development for Beginners* de Google.

Issue	Estimado	Final
Documentar los resultados de las nuevas ejecuciones en el cuaderno de investigación	8	8
Aprender a usar Android Studio	13	21

Tabla A.18: Tareas del sprint 18

Sprint 19

Fecha: 02/05/2019 - 09/05/2019

Se terminó el comportamiento de la pantalla de autenticación, se generó el clasificador obtenido con el mejor conjunto de características encontrado y se continuó con la documentación de la planificación temporal.

Issue	Estimado	Final
Terminar el comportamiento de la pantalla de login	3	3
Extraer características deseadas con tsfresh y generar clasificador	5	5
Documentación de la memoria	3	3

Tabla A.19: Tareas del sprint 19

Sprint 20

Fecha: 02/05/2019 - 16/05/2019

Se crearon las pantallas principales de la aplicación de Android.

<i>Issue</i>	Estimado	Final
Crear la pantalla de visualización de camas	8	13
Crear pantalla de visualización de datos	13	13
Terminar pantallas de gestión de Usuarios	5	5

Tabla A.20: Tareas del sprint 20

Sprint 21

Se terminó el comportamiento general de la aplicación de Android, se solucionaron algunos bugs y se adecuaron las interfaces de usuario a la guía de estilos empleada.

Fecha: 16/05/2019 - 23/05/2019

<i>Issue</i>	Estimado	Final
Terminar comportamiento de la pantalla de visualización de datos	5	5
Pantallas de gestión de camas	8	8
Modificar estructura de las pantallas de gestión de usuarios	5	5
Bug en la visualización de los datos	3	3
Modificaciones menores de las interfaces para adecuarse a la guía de estilos	5	5
Crear menú para el rol de usuario	8	13

Tabla A.21: Tareas del sprint 21

Sprint 22

Fecha: 23/05/2019 - 30/05/2019

Se codificó el comportamiento de la aplicación ante pérdidas de conexión y ante pérdidas de la sesión por autenticación con el mismo usuario en otro dispositivo.

<i>Issue</i>	Estimado	Final
Comprobar si la sesión ha caducado y redirigir	3	3
Controlar la pérdida de conexión a internet	5	5

Tabla A.22: Tareas del sprint 22

Sprint 23

Fecha: 30/05/2019 - 07/06/2019

Se solucionaron varios bugs de la aplicación.

<i>Issue</i>	Estimado	Final
Bug en la pantalla de gráficas	5	5
Bug en la pantalla de administración cuando la lista supera la longitud de la pantalla	5	5
Cambiar cambio de contraseña de usuarios desde el administrador	2	2

Tabla A.23: Tareas del sprint 23

Sprint 24

Fecha: 07/06/2019 - 13/06/2019

Se comenzó a redactar la memoria y se eliminaron algunos bugs de la aplicación.

<i>Issue</i>	Estimado	Final
Comenzar documentación de la memoria	13	13
Bug de las gráficas al refrescar	8	8

Tabla A.24: Tareas del sprint 24

Sprint 25

Fecha: 13/06/2019 - 19/06/2019

Se terminó la primera versión de la memoria, se generó el archivo apk definitivo y se probó en distintos dispositivos compatibles.

<i>Issue</i>	Estimado	Final
Eliminar filas huérfanas y viudas	-	1
Terminar la primera versión de la memoria	13	21
Generar apk y probarla en varios dispositivos	3	3
Eliminar errores al instalar en otros dispositivos	5	5

Tabla A.25: Tareas del sprint 25

Sprint 26

Fecha: 19/06/2019 - 27/06/2019

Se añadió una pantalla «about» a la aplicación, se comentó y documentó el código usando javadoc y se terminó la documentación de los anexos.

<i>Issue</i>	Estimado	Final
Añadir pantalla de «about»	5	5
Comentar código	5	5
Generar javadoc	3	1
Terminar los anexos	21	21
Generar test con Espresso	5	5
Mejorar el README	1	1

Tabla A.26: Tareas del sprint 26

Coste total de cada sprint

En la tabla A.27 se muestran los costes estimados y finales totales de cada sprint y la suma del coste final del proyecto.

<i>Sprint</i>	Estimado	Final
Sprint 1	15	15
Sprint 2	22	22
Sprint 3	21	26
Sprint 4	21	34
Sprint 5	26	29
Sprint 6	31	31
Sprint 7	24	16
Sprint 8	37	39
Sprint 9	13	16
Sprint 10	8	16
Sprint 11	13	13
Sprint 12	18	18
Sprint 13	23	15
Sprint 14	26	34
Sprint 15	21	21
Sprint 16	20	28
Sprint 17	16	16
Sprint 18	21	29
Sprint 19	11	11
Sprint 20	26	31
Sprint 21	34	39
Sprint 22	8	8
Sprint 23	12	12
Sprint 24	21	21
Sprint 25	21	30
Sprint 26	40	38
Total	549	608

Tabla A.27: Coste de cada sprint.

A.3. Estudio de viabilidad

Viabilidad económica

Para considerar la viabilidad económica del proyecto se deben calcular los costes derivados de su realización. Se van a tener en cuenta tanto el coste del personal como el del softwares y el hardware empleados. Dado que este proyecto se ha realizado de forma conjunta con José Luis Garrido

Labrador, ambos calcularemos el coste asumiendo que el proyecto cuenta con dos empleados.

Costes de personal

Siguiendo estas consideraciones calculamos el coste total de personal de la siguiente forma:

Concepto	Coste(€)
Salario mensual neto [6]	1 225,7
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.28: Costes de personal.

Costes del *software*

El *software* empleado no supone ningún coste en este proyecto ya que todas las herramientas y bibliotecas utilizadas son de código abierto o gratuitas.

Costes del *hardware*

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 coste de amortización correspondiente a la duración del proyecto (7 meses):

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.29: Costes de *hardware*.

Coste total

Teniendo en cuenta los costes de personal y de *hardware*, el coste económico total del proyecto asciende a:

Concepto	Coste(€)
Coste de personal	28 156,24
Coste del <i>hardware</i>	985,83
Total	29 142,07

Tabla A.30: Coste total.

Viabilidad legal

En esta sección se hablará de la viabilidad legal del proyecto en términos de la licencia del software generado. A la hora de escoger la licencia más adecuada para nuestro software estamos limitados por las condiciones de las licencias de las herramientas o bibliotecas que hemos empleado para generarlos.

En la aplicación se han usado herramientas y bibliotecas con las siguientes licencias ordenadas de más a menos permisivas:

- MIT: Socket.IO-client Library.
- Apache 2.0: Android Studio, Gradle, Android Support Library, MPAndroidChart.

Para la fase de investigación y documentación se usan herramientas y bibliotecas con las siguientes licencias ordenadas de más a menos permisivas:

- MIT: tsfresh.
- Zero clause BSD: tmux.
- Modified BSD: Anaconda, Scikit-Learn, Jupyter Notebook.
- GPLv2: Pencil.
- GPLv3: Weka, DEAP.

Todas las licencias son compatibles con GPL, siendo GPLv3 la más restrictiva. Para este proyecto se ha decidido usar la licencia AGPLv3[14] (*GNU Affero General Public License v3*), derivada de GPLv3 pero diseñada específicamente para asegurar la cooperación con la comunidad en el caso de software que corra en servidores de red.

Apéndice B

Especificación de Requisitos

B.1. Introducción

Aunque este trabajo se centra sobre todo en investigación, cuando hablamos de requisitos nos referiremos a los de la aplicación Android generada. En esta sección se enumerarán los requisitos funcionales y no funcionales de la aplicación, y se definirán los casos de uso derivados.

B.2. Objetivos generales

En la memoria se exponen los objetivos generales del trabajo, los cuales, dada la naturaleza del trabajo, se centran principalmente en la fase de investigación. En este apartado nos centraremos en los requisitos relativos al último de los objetivos generales expuestos:

«Desarrollar una app de Android para mostrar la aplicabilidad del modelo de clasificación generado.»

B.3. Catalogo de requisitos

Aquí se enumeran los requisitos funcionales y no funcionales de la aplicación desarrollada para dispositivos Android. Dado que mi compañero de proyecto José Luis Garrido Labrador y yo hemos realizado dos aplicaciones (una web y una para Android) con el mismo objetivo y las mismas funcionalidades, la especificación de los requisitos se ha realizado de forma conjunta, y por lo tanto, muchos de los puntos de estos apartados coincidirán en ambos trabajos.

Requisitos funcionales

- **RF-1 Confidencialidad del sistema:** Solamente los 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 los datos de todas las camas existentes.
- **RF-2 Gestión de las camas:** El administrador debe poder gestionar las camas pudiendo añadir, modificar, borrar y dar acceso a un usuario a los datos de una cama determinada.
 - **RF-2.1 Añadir cama:** el administrador debe poder añadir una nueva cama al sistema.
 - **RF-2.2 Modificar cama:** el administrador debe poder modificar los datos una cama existente.
 - **RF-2.3 Borrar cama:** el administrador debe poder borrar una cama del sistema.
 - **RF-2.4 Asignar camas a usuarios:** el administrador se encarga de decidir qué usuario puede acceder a los datos de qué cama.
- **RF-3 Gestión de los usuarios:** el administrador debe poder gestionar los usuarios pudiendo añadir, modificar y borrar. El usuario debe poder gestionar su propia contraseña.
 - **RF-3.1 Añadir usuario:** el administrador debe poder añadir un nuevo usuario al sistema.
 - **RF-3.2 Modificar usuario:** el administrador debe poder modificar los datos un usuario existente. Igualmente el usuario debe poder modificar su propia contraseña.

- **RF-3.3 Borrar usuario:** el administrador debe poder borrar un usuario del sistema.
- **RF-4 Visualización de los datos:** los usuarios deben poder ver, de las camas disponibles, el estado actual del paciente, la probabilidad de crisis epiléptica, 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 Confidencialidad:** los datos de las camas, al ser en parte constantes vitales de pacientes, solamente han de ser accesibles por los usuarios permitidos.
- **RNF-3 Escalabilidad:** el sistema debe ser escalable para adaptarse de manera correcta a un incremento de carga del sistema.
- **RNF-4 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.

B.4. Especificación de requisitos

De la misma forma, en lo relativo a las funcionalidades del cliente, la especificación de los casos de uso se ha hecho de forma conjunta con mi compañero José Luis Garrido Labrador, por lo que los contenidos de este apartado coincidirán en gran medida con los suyos.

Actores

En los casos de uso se distinguen dos actores:

- **Administrador:** Tiene acceso a la gestión de usuarios, la gestión de camas y la visualización de los datos de todas las camas existentes.
- **Usuario:** Tiene acceso a la visualización de los datos de las camas que tiene asignadas y a la gestión de su propio usuario.

Diagramas de casos de uso

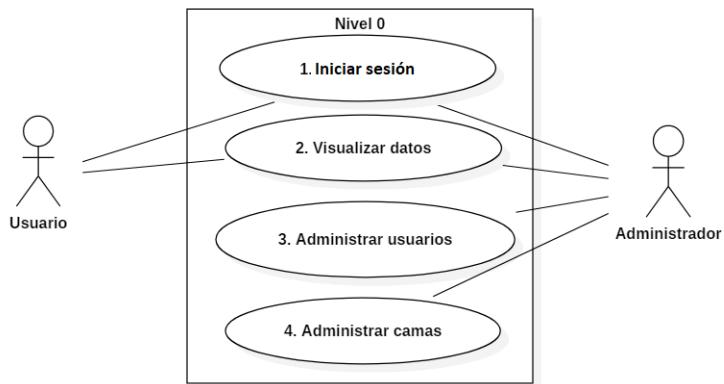


Figura B.1: Diagrama de casos de uso, nivel 0.

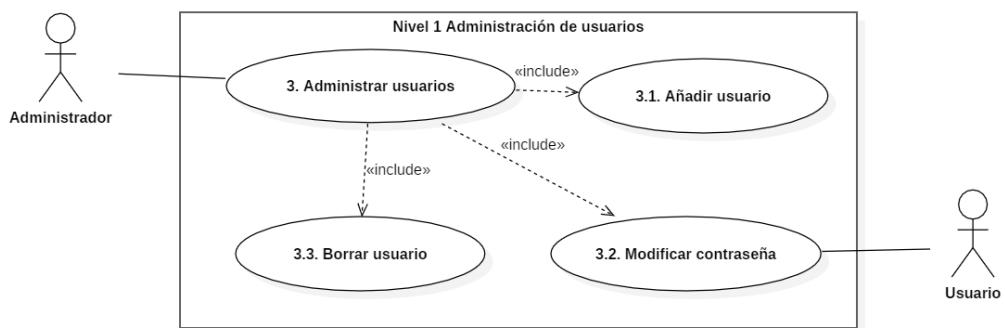


Figura B.2: Diagrama de casos de uso, visualización de datos.

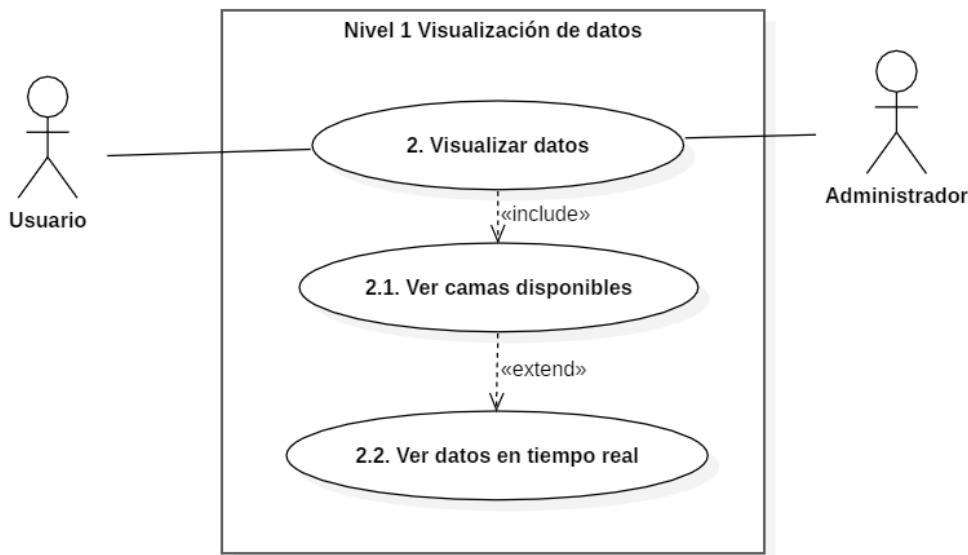


Figura B.3: Diagrama de casos de uso, administración de usuarios.

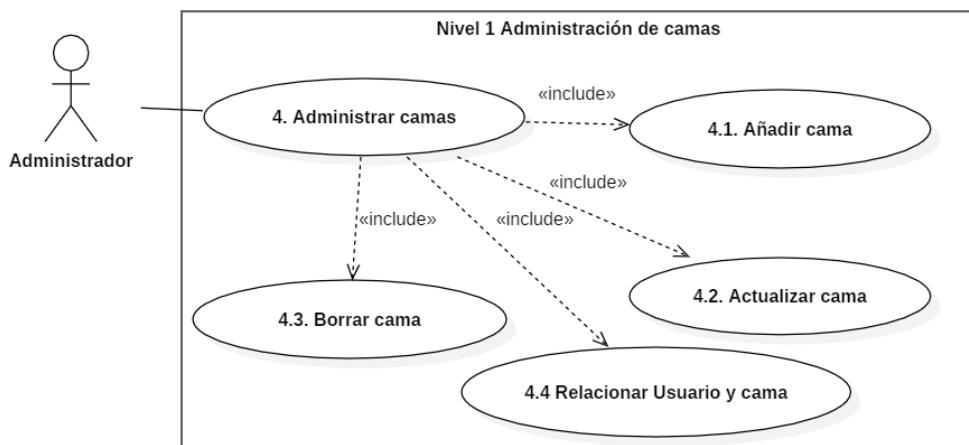


Figura B.4: Diagrama de casos de uso, administración de camas.

Especificación de casos de uso

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 genérico 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	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;">1</td><td>Elegir cama</td></tr> <tr> <td>2</td><td>Elegir usuario</td></tr> <tr> <td>3</td><td>Si la relación existe se puede eliminar el permiso</td></tr> <tr> <td>3</td><td>Si la relación no existe se puede crear el permiso</td></tr> </table>	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
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

En esta sección se va a hablar de los aspectos más relevantes del diseño de la aplicación de Android.

- En primer lugar, se hará una breve mención al diseño de los datos.
- A continuación, en el apartado de diseño procedimental se expondrán detalladamente los dos procesos más relevantes de la aplicación, y en los que se basa fundamentalmente su comportamiento, obviando otros procesos más sencillos o basados en objetos propios del *framework* de Android cuyo funcionamiento puede encontrarse en la documentación.
- Después se expondrán las características de la arquitectura general de la aplicación, considerando dos puntos de vista que pueden servir para definir de forma complementaria la arquitectura general del sistema.
- Por último se expondrán los prototipos de la interfaz de usuario generados antes del desarrollo de la aplicación.

C.2. Diseño de datos

Para el desarrollo de la aplicación Android no podemos hablar estrictamente de un diseño de los datos, ya que se trata de un cliente que no interactúa directamente con la base de datos. El acceso a la persistencia se hace a través de peticiones a la API del servidor, la cual devuelve siempre todos los datos en un objeto con formato JSON.

En este sentido, podemos abstraernos del diseño e implementación concretos de la base de datos, ya que nuestra gestión de los datos solo depende de las características del JSON que nos va a devolver cada petición. La estructura de la respuesta de cada petición a la API se especifica en el Manual del programador de mi compañero José Luis Garrido Labrador.

C.3. Diseño procedimental

Como se explica más adelante, el funcionamiento general de la aplicación se basa en la comunicación con la API del servidor remoto, el cual proporciona el modelo de datos y la lógica de negocio. Por esta razón, los procesos más relevantes corresponden con los que permiten realizar la comunicación de la aplicación con la API del servidor remoto.

Tal y como se ha planteado la aplicación de Android, existen dos procesos relevantes que es importante entender, ya que supondrán la base del funcionamiento de la aplicación:

- En primer lugar, la **llamada a los servicios genéricos de la API**. Todas las interacciones con la lógica de negocio de la aplicación (iniciar sesión, visualizar usuarios, añadir usuarios...) se realizan de esta manera. Por esta razón, para garantizar el buen funcionamiento de la aplicación, este proceso debe incluir una comprobación sobre si existe conexión a internet, si la sesión está activa y si la respuesta de la API es la esperada.

Esta interacción se representa en el diagrama de secuencias de la figura C.1. El elemento *Activity* corresponde con cualquiera de las clases que extienden de *AppCompatActivity* desde las que se realiza una petición a la API del servidor.

- En segundo lugar, la **petición y recepción de los datos de una cama en tiempo real**. Este proceso se realiza a través de conexión a nivel de *sockets* con el servidor y la gestión de eventos empleando la librería Socket.IO.

Esta interacción se representa en el diagrama de secuencias de la figura C.2. Los mensajes `give_me_data` y `package` son eventos de Socket.IO definidos por el programador de la API.

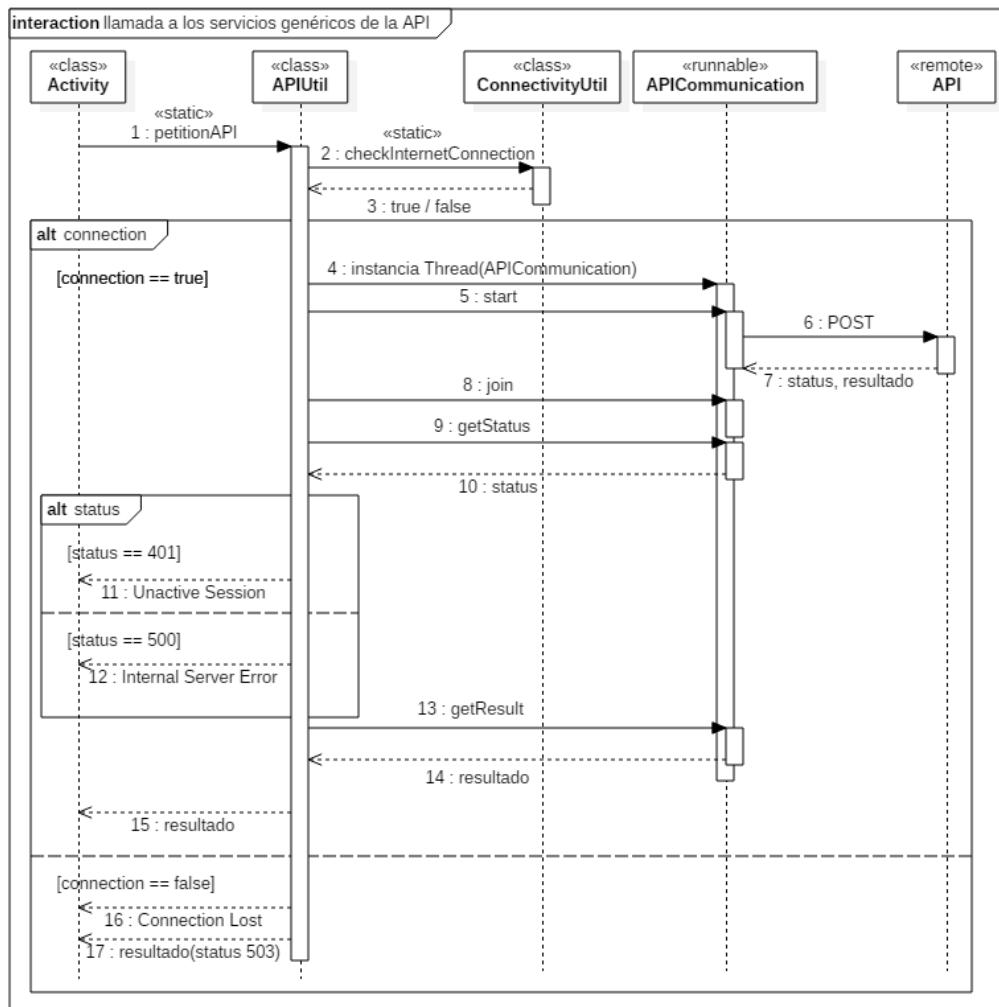


Figura C.1: Diagrama de secuencias, llamada a los servicios genéricos de la API.

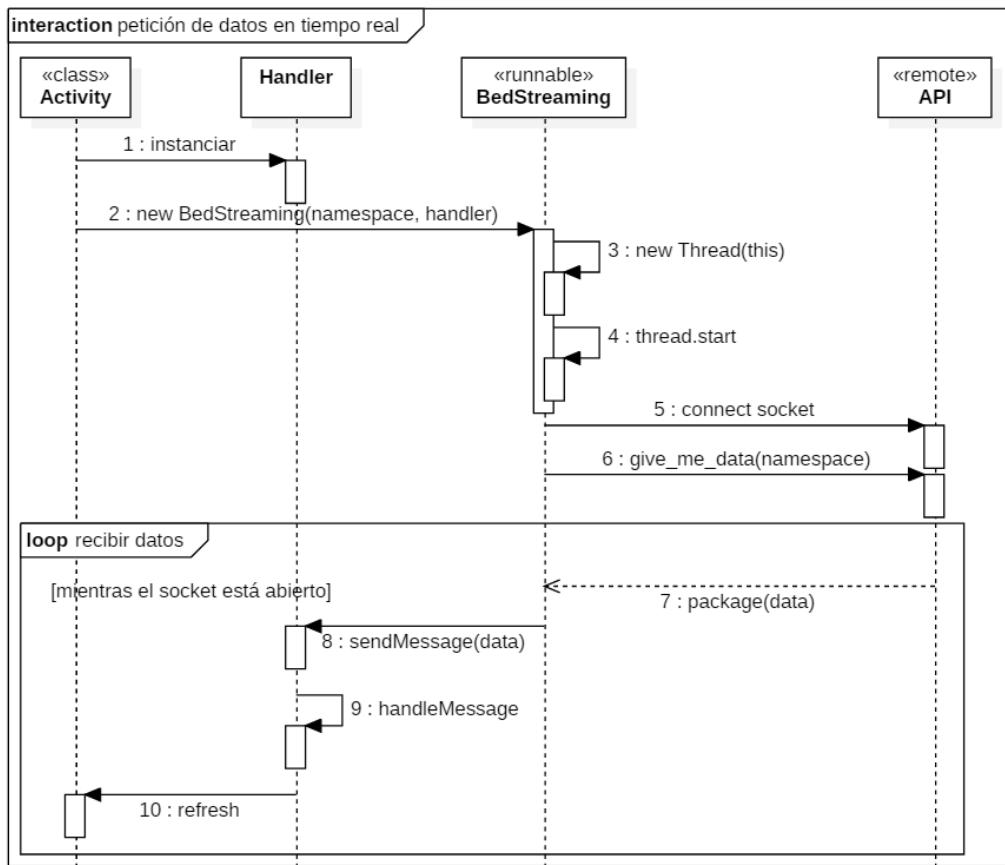


Figura C.2: Diagrama de secuencias, petición de datos en tiempo real.

El funcionamiento del resto de procesos es fácilmente deducible a partir del código o corresponde con el uso de elementos propios del *framework* de Android. La especificación y funcionamiento de estos elementos se puede consultar en la página web oficial de *Android Developers* [1].

C.4. Diseño arquitectónico

La arquitectura de la aplicación puede verse desde varios puntos de vista. En el contexto del patrón arquitectónico Modelo-Vista-Controlador (MVC) [15] definimos tres componentes:

- **Modelo:** Contiene la estructura de los datos que maneja el programa e interactúa con la persistencia para realizar consultas y actualizaciones.

- **Vista:** Presenta los datos del modelo y la interacción con la lógica de negocio de forma adecuada para el usuario.
- **Controlador:** Responde a eventos generados por la interacción del usuario con la vista y realiza peticiones de consulta/actualización al modelo. Se puede ver como un intermediario entre ambos componentes.

Según la definición de los componentes de este patrón, la aplicación Android podría ser considerada únicamente como un componente «Vista» de un sistema más grande formado también por los servicios proporcionados por la API del servidor remoto, la cual proporciona el acceso a la persistencia y la comunicación entre el modelo de datos y la aplicación.

Por otro lado, cuando se habla de este tipo de aplicaciones, es común encontrarnos con el término **«arquitectura de microservicios»**. Según esta arquitectura, cada funcionalidad se encuentra contenida en un proceso del servidor al que llamaremos microservicio. Cada microservicio encapsula un solo aspecto de la lógica de negocio, y los microservicios son procesos independientes entre sí. Tal y como ocurre en esta aplicación Android, el cliente se limita a hacer peticiones a los microservicios de la API del servidor remoto, los cuales, basándose en la lógica de negocio, le proporcionan los datos necesarios para actualizar la «vista» o interfaz de usuario.

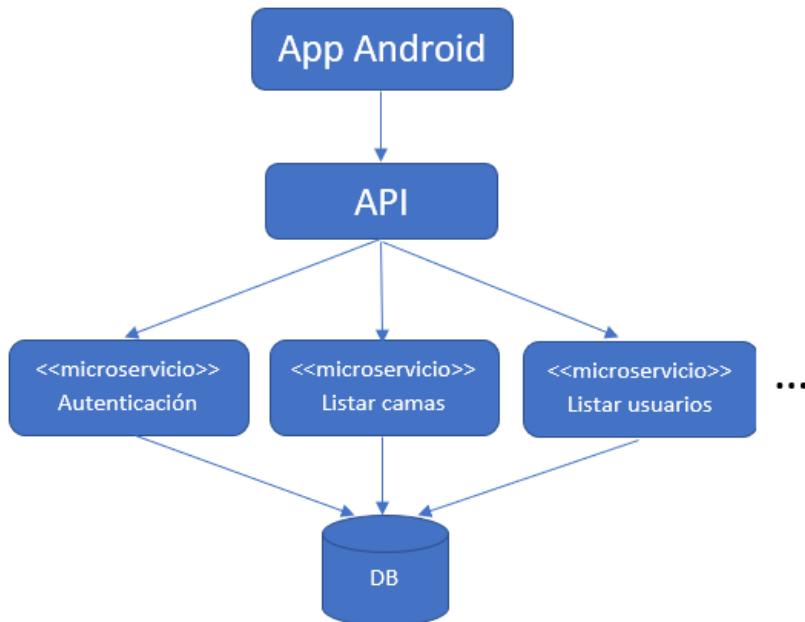


Figura C.3: Abstracción de la arquitectura de microservicios.

C.5. Diseño de interfaces

Inicialmente se realizaron una serie de prototipos básicos en los que se plasmaron las principales funcionalidades de la aplicación, sin prestar especial atención a los aspectos estéticos de la misma. Para ello se usó la herramienta de prototipado Pencil, ya que permite incorporar elementos propios de la guía de estilos que se ha seguido para el diseño de las interfaces de usuario: *Material Design* [4].

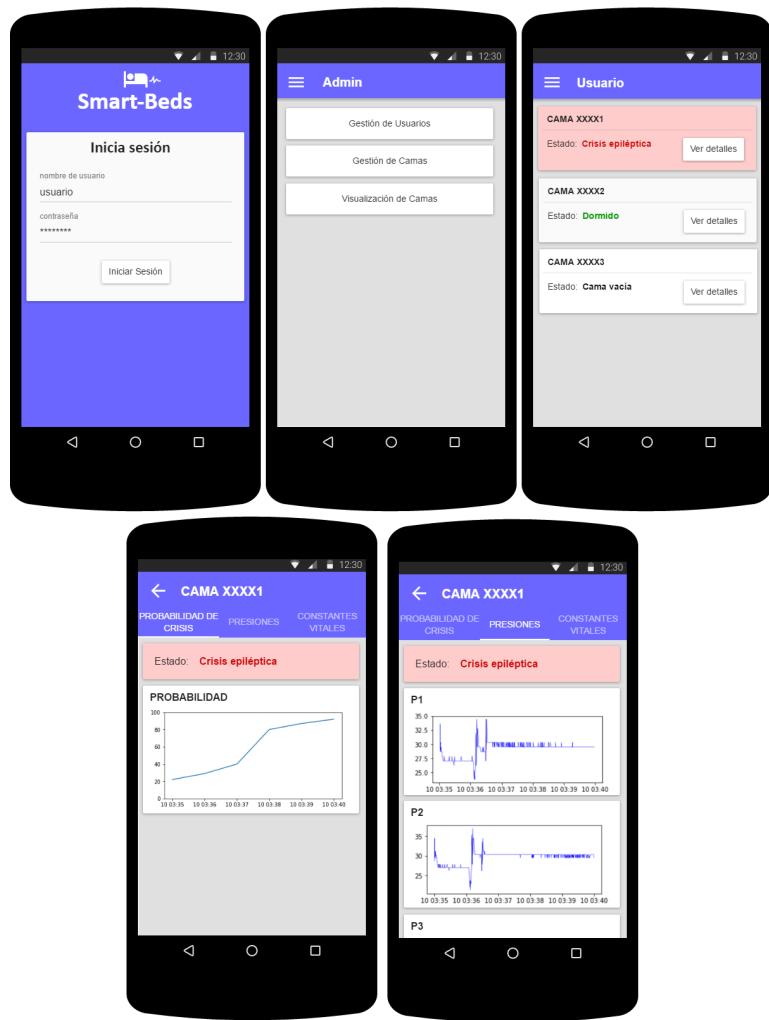


Figura C.4: Prototipos iniciales de las pantallas de: login, administración, visualización de camas y visualización de datos.

Tal y como se recomienda en la guía de estilos y como se muestra en la figura C.5, se escogieron dos colores principales para la interfaz: un color primario y un color secundario, más las consiguientes derivados de los mismos.

- El color primario y sus derivados se emplearon en las barras de herramientas, los botones y el menú de navegación.
- El color secundario y su derivado se empleó para enfatizar elementos en los que el usuario debía realizar una acción, o elementos a los que debe prestar especial atención (como una cama en estado de crisis epiléptica).

Además, para dar una sensación de claridad y limpieza, se decidieron mantener los fondos blancos, y el color del texto se estableció como claro (blanco) u oscuro (negro/gris) dependiendo del color del elemento donde se encontraba. De este modo se genera un contraste suficiente para que el texto resulte legible y claro.

Los tamaños y separaciones entre textos y el resto de elementos se establecieron según las recomendaciones de la guía de estilos.

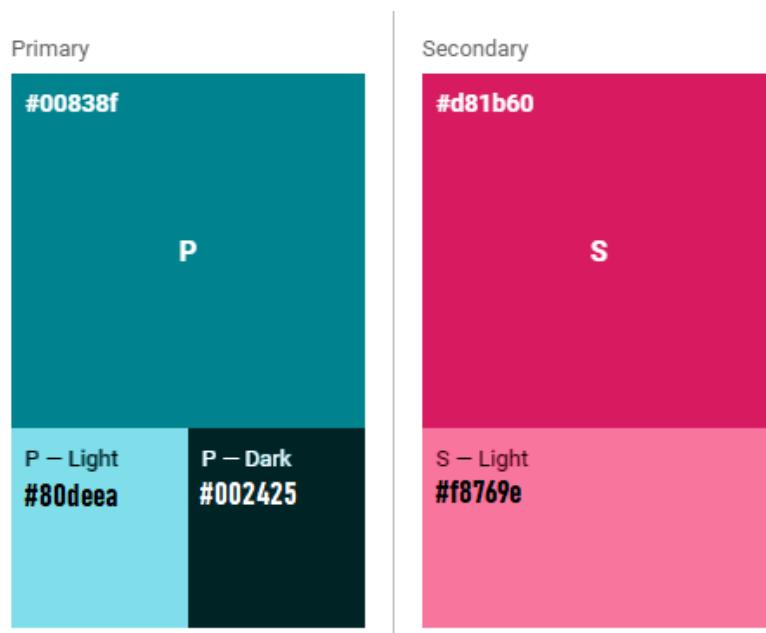


Figura C.5: Paleta de colores.

Durante el proceso de desarrollo se tomaron varias decisiones de diseño cuyo resultado fueron las interfaces de usuario finales que se muestran en la figura C.6

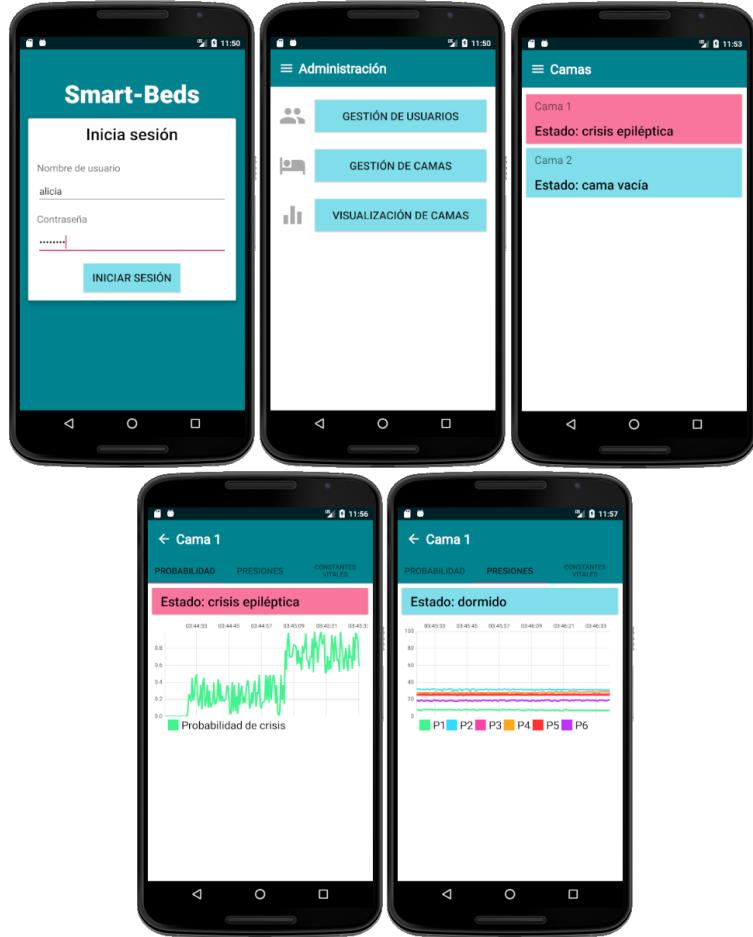


Figura C.6: Interfaces de usuario finales de las pantallas de: login, administración, visualización de camas y visualización de datos.

Apéndice D

Documentación técnica de programación

D.1. Introducción

En este apartado se van a exponer todos los conceptos necesarios para comprender la estructura de proyecto, instalar el software necesario para su integración, importarlo en un nuevo equipo, compilarlo, ejecutarlo y exportar la aplicación.

Además, se exponen las herramientas que se han usado para generar las pruebas del software, se explica cómo usarlas para generar nuevas pruebas y cómo ejecutar las pruebas existentes.

D.2. Estructura de directorios

En la figura D.1 se muestra la estructura de directorios del repositorio de GitHub en el que se encuentra alojado el proyecto: <https://github.com/aog0036/TFG-SmartBeds>.

El contenido del repositorio se estructura principalmente en tres directorios:

- **/android:** Contiene el proyecto de Android Studio con los ficheros de código fuente, los de test, los de configuración y el fichero .apk generado.
- **/doc:** Contiene la documentación general del proyecto, incluyendo la memoria, los anexos y el cuaderno de investigación, todos en formato

APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN

.pdf y en formato .tex. También contiene otros recursos como las imágenes y los archivos con las referencias bibliográficas (extensión .bib).

- **/jupyter notebooks:** Contiene los notebooks y los scripts de python con los experimentos generados durante la fase de investigación.

Dentro del proyecto de Android Studio los directorios más importantes son los siguientes:

- **/app:** Contiene los directorios release, src y los que contienen los ficheros de pruebas unitarias y de interfaz.
 - **/release:** Contiene el fichero SmartBeds.apk para la instalación y distribución de la aplicación en un dispositivo Android.
 - **/src/main/java/.../smartbeds:** Contiene los paquetes con las clases Java que componen el código fuente de la aplicación.
 - **/src/main/res:** Contiene los directorios con los archivos .xml que definen los recursos gráficos de la aplicación (interfaces, colores, menús...).
 - **/src/main/AndroidManifest.xml:** Manifiesto que contiene información esencial sobre la aplicación y que permite al sistema ejecutarla.
 - **/src/androidTest/.../generalActivities:** Contiene las pruebas de interfaz gráfica generadas mediante la herramienta Espresso.
 - **/src/test/.../smartbeds:** Contiene las pruebas unitarias generadas mediante JUnit.
- **/javadoc:** Contiene la documentación de las clases y métodos del código de la aplicación en formato html.
- **archivos de configuración de gradle:** Son una serie de directorios y ficheros generados automáticamente por Android Studio y que permiten construir la aplicación con la configuración y las dependencias necesarias.

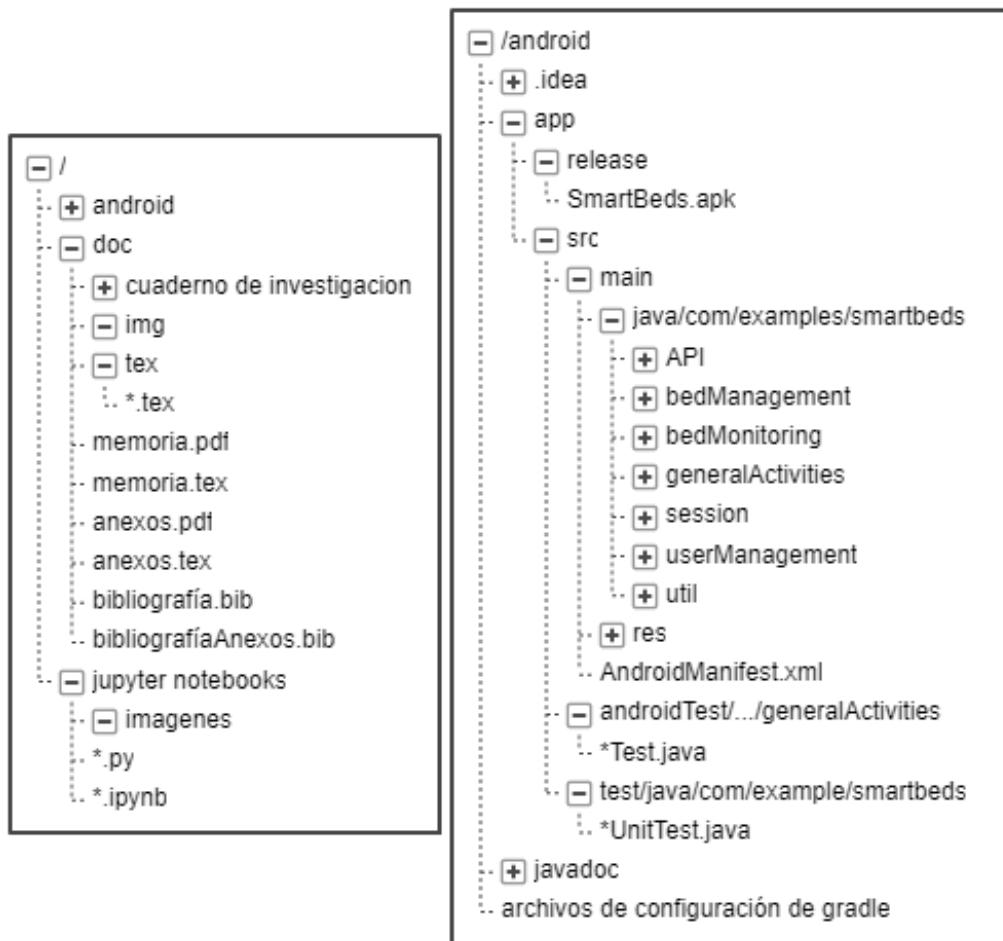


Figura D.1: Estructura de directorios del repositorio.

D.3. Manual del programador

Experimentos

En el repositorio no se incluye el directorio /data con los ficheros .csv ya que estos datos son propiedad de la empresa proveedora de los datos. Estos datos no son necesarios para poder visualizar los procesos que se han seguido en cada experimento, los distintos métodos que se han empleado y los resultados conseguidos (para ello es necesario tener instalada la herramienta Jupyter Notebook). En todo caso, en el CD proporcionado al tribunal se incluye un enlace a una carpeta en One Drive en el archivo **data_link** pero se debe tener en cuenta que estos datos no podrán ser distribuidos ni empleados

APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN

para usos ajenos a este trabajo. La carpeta `/data` debe encontrarse en el directorio raíz para poder ejecutar los experimentos.

En caso de contar con el directorio `/data` que contiene los datos proporcionados por el proveedor, se recomienda instalar Anaconda, ya que además de incluir Jupyter Notebook, instala por defecto muchas de las bibliotecas que se emplean en los experimentos. Todos los experimentos se han escrito en el lenguaje de programación **Python 3**.

Aplicación Android

Para trabajar con el proyecto de Android Studio es necesario tener instalado **Java JDK 8** y **Android Studio**. En este caso se ha usado una versión de Android Studio 3.3.2. El resto de dependencias se encuentran indicadas en los ficheros de configuración de gradle, y se añaden automáticamente al construir la aplicación. Concretamente las dependencias de la aplicación se encuentran en el fichero `/android/app/build.gradle`, y será aquí donde se deberán añadir las dependencias nuevas si son necesarias. En este archivo también se indica la versión mínima de la API de Android que se soporta, en este caso la 23, que corresponde con la versión 6 de Android (*Marshmallow*). Esto supone que la aplicación funcionará en dispositivos con una versión 6 de Android o superior.

Además, el funcionamiento de esta aplicación depende de los servicios ofrecidos por la API implementada por mi compañero José Luis Garrido Labrador. Los requisitos e instalación de la API del servidor remoto se encuentran documentados en los anexos de su trabajo: <https://github.com/jlgarridol/TFG-SmartBeds>.

D.4. Compilación, instalación y ejecución del proyecto

En este apartado se indica cómo importar el proyecto de Android Studio, cómo ejecutar la aplicación y cómo exportar el archivo .apk para la distribución en instalación de la aplicación en un dispositivo Android.

Importar el proyecto

Una vez instalados Java JDK 8 y Android Studio, para importar el proyecto se deben seguir los siguientes pasos:

D.4. COMPILACIÓN, INSTALACIÓN Y EJECUCIÓN DEL PROYECTO⁴⁵

1. Acceder al repositorio: <https://github.com/aog0036/TFG-SmartBeds>.
2. Descargar el contenido del repositorio desde **Clone or download** > **Download ZIP**.

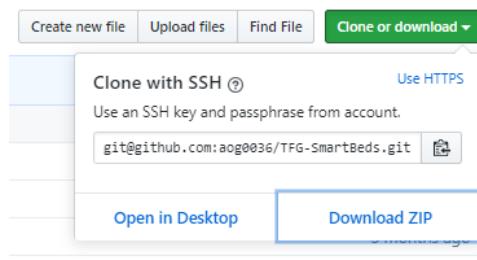


Figura D.2: Descarga del contenido del repositorio.

3. Descomprimir el fichero .zip en la ruta en la que se desee alojar el proyecto.
4. Abrir Android Studio.
5. En el menú de opciones de Android Studio ir a **File > Open** y seleccionar el directorio **/android**.

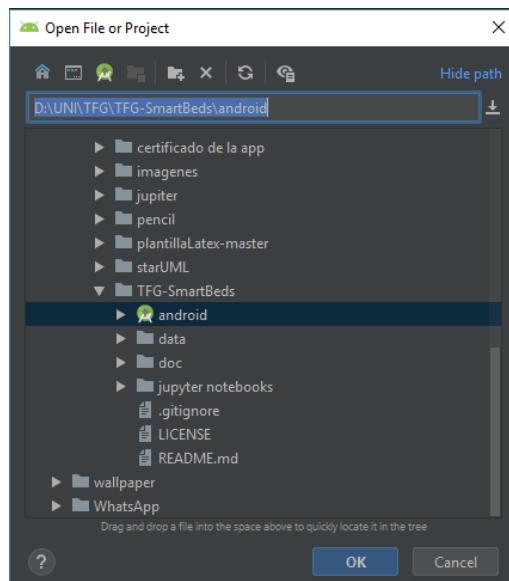


Figura D.3: Importar proyecto de Android Studio.

APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN

Ejecutar la aplicación

Existen dos formas de ejecutar la aplicación en Android Studio: conectando un dispositivo Android mediante USB o usando un emulador. Para que tu equipo reconozca el dispositivo Android al conectarlo, es necesario que tengas instalados sus *drivers*, por lo demás, es la opción más rápida y cómoda. Por otro lado, si no se dispone de un dispositivo que tenga una versión de Android compatible con la aplicación, puedes crear un nuevo emulador de la siguiente forma:

1. En el menú de opciones ir a **Run > Run ‘app’**.
2. En la ventana emergente que se muestra seleccionar la opción **Create New Virtual Device**.
3. Aquí podrás escoger el modelo del dispositivo que quieras emular y la versión de Android a instalar.

Para ejecutar la aplicación, tanto en un dispositivo conectado al equipo como en el emulador, se siguen los siguientes pasos:

1. En el menú de opciones ir a **Run > Run ‘app’**.
2. En la ventana emergente seleccionar el dispositivo que quieras usar y pulsar **OK**.

Exportar apk

Para exportar el archivo .apk que permite distribuir e instalar la aplicación en un dispositivo Android se siguen los siguientes pasos:

1. En el menú de opciones ir a **Build > Build Bundle(s) / APK(s) > Build APK(s)**.
2. El archivo apk generado se encontrará en el directorio `/android/app/build/outputs/apk/debug`.

Si se desea generar un apk para subirlo a la plataforma Google Play se debe escoger la opción **Build > Generate Signed Build / APK...** que crea una apk firmada, asociada a un certificado que es necesario para distribuir la aplicación en Google Play.

D.5. Pruebas del sistema

Además de las pruebas manuales, se han realizado dos tipos de pruebas del sistema: pruebas unitarias y pruebas de sistema, más concretamente de interfaz de usuario.

Pruebas unitarias

Para realizar las pruebas unitarias sobre las clases que no están directamente asociadas con una interfaz de usuario se ha empleado la biblioteca de pruebas unitaria sobre Java **JUnit**. Las pruebas unitarias comprueban el buen comportamiento de un solo módulo o clase de forma aislada.

Estas pruebas se encuentran en el directorio
`/android/app/src/test/java/com/example/smartbeds.`

Prueba	Descripción
BedUnitTest.java	Prueba el funcionamiento de los <i>getters</i> y <i>setters</i> de la clase Bed.
SessionUnitTest.java	Prueba el comportamiento <i>singleton</i> de la clase Session, el funcionamiento de sus <i>getters</i> y <i>setters</i> y la función <code>resetSession</code> .

Tabla D.1: Pruebas unitarias.

Pruebas de interfaz

Además de las pruebas unitarias, se han realizado una serie de pruebas sobre la interfaz de usuario mediante la herramienta **Espresso**. Esta herramienta permite grabar las acciones del usuario sobre los elementos de la interfaz de usuario, hacer comprobaciones del contenido tras cada acción y volver a ejecutar la grabación. Este tipo de pruebas resultan más útiles para una aplicación pequeña como esta.

Para ejecutar las pruebas de interfaz debemos seleccionar el directorio que las contiene en el menú de navegación del proyecto y pulsar **Run ‘Tests in ‘nombre del directorio’’**.

Además, la herramienta Espresso se encuentra completamente integrada en Android Studio, permitiendo generar una nueva prueba de la siguiente forma:

APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN

1. En el menú de opciones de Android Studio ir a **Run > Record Espresso Test**.
2. Escoger el dispositivo (real o emulado) sobre el que se desea grabar la prueba y pulsar **OK**.
3. Se abrirá una ventana emergente en la que se irán registrando las acciones.
4. Tras cada acción, pulsando en la opción **Add Assertion** de la ventana emergente se pueden añadir comprobaciones sobre el contenido de la interfaz (por ejemplo, comprobar que cierto elemento está presente o que cierto *TextView* contiene un texto determinado).
5. Una vez realizada la grabación al pulsar **OK** se generará el código correspondiente de la prueba con el nombre que se le indique.

Para ejecutar la prueba generada debemos seleccionarla en el menú de navegación del proyecto con click derecho y seleccionar **Run ‘nombre del test’**. También podemos ejecutar todas las pruebas seleccionando con click derecho el directorio que las contiene y pulsando **Run ‘Tests in ‘nombre del directorio’’**.

Estas pruebas se encuentran en el directorio
`/android/app/src/androidTest/java/com/example/smartbeds/generalActivities`.

Prueba	Descripción
loginTest.java	Prueba el correcto funcionamiento de la pantalla de inicio de sesión.
changePassTest.java	Prueba el correcto funcionamiento de la pantalla de cambio de contraseña.
addDeleteUserTest.java	Prueba las funciones añadir y eliminar usuario, comprobando que un usuario eliminado no puede acceder al sistema.
asignUserBedTest.java	Prueba que la asignación de una cama a un usuario funciona correctamente.
navigationTest.java	Prueba el correcto funcionamiento del menú de navegación y todas sus opciones.

Tabla D.2: Pruebas de interfaz.

Apéndice E

Documentación de usuario

E.1. Introducción

En esta sección se enumeran los requisitos que debe cumplir el usuario para poder usar la aplicación, se explica el proceso de instalación y se ofrece un manual de usuario completo para aprender a usarla.

E.2. Requisitos de usuarios

Para poder instalar y usar la aplicación el dispositivo del usuario debe cumplir los siguientes requisitos:

- Contar con una versión de Android 6 (*Marshmallow*) o superior, la cual corresponde con la API 23 de Android.
- Permitir la instalación de aplicaciones de orígenes desconocidos [5]. Para ello:
 1. Ir a los ajustes del dispositivo.
 2. A la opción de **Seguridad** o **Privacidad** según la versión.
 3. Activar la opción **Orígenes desconocidos**.
- Contar con conexión a internet.

Además, para poder entrar en la aplicación deberá tener un usuario y una contraseña asignados. Para que el tribunal pueda acceder al sistema se proporcionan dos cuentas de usuario: una con rol de administrador y otra con rol de usuario genérico:

- **Administrador:**

- Nombre de usuario: **admin**
- Contraseña: **admin**

- **Usuario genérico:**

- Nombre de usuario: **tfg**
- Contraseña: **tfg**

E.3. Instalación

La instalación de una aplicación Android es muy sencilla. Solo es necesario pasar el archivo **SmartBeds.apk** al sistema de ficheros del dispositivo Android y seleccionarlo. En caso de tener activada la opción de permitir la instalación de aplicaciones de orígenes desconocidos y contar con el suficiente espacio de almacenamiento la aplicación, se instalará automáticamente.

Se puede acceder al archivo .apk desde el *release* 1.0 del repositorio del proyecto¹, o en la ruta /android/app/release.

E.4. Manual del usuario

En esta sección se ofrece un manual para el uso sobre todas las funcionalidades de la aplicación.

Iniciar sesión

Al ejecutar la aplicación desde el dispositivo Android la primera pantalla que se muestra es la de «**Iniciar Sesión**». En esta pantalla se nos pide un nombre de usuario y una contraseña para acceder al sistema. Se debe:

1. Introducir el nombre de usuario.
2. Introducir la contraseña.
3. Pulsar el botón «Iniciar sesión».

¹<https://github.com/aog0036/TFG-SmartBeds>

En caso de haber introducido valores correctos se accederá a la siguiente pantalla, en caso contrario un cuadro de diálogo nos indicará que el usuario o la contraseña introducidos no son correctos.

Si el usuario introducido tiene un rol de administrador la siguiente pantalla será el «**Menú de administración**». Si tiene un rol de usuario la siguiente pantalla será la de «**Visualización de camas**».

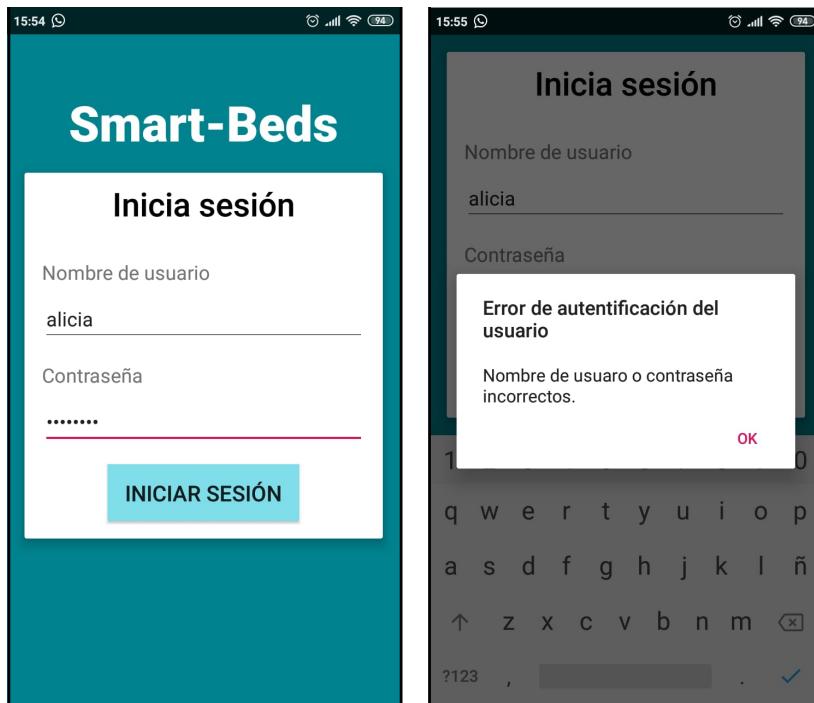


Figura E.1: Iniciar sesión.

Menú de administración

Este menú solo está disponible si el usuario tiene un rol de administrador. En él se puede escoger a qué función de la aplicación deseas acceder: «**Gestión de usuarios**», «**Gestión de camas**» o «**Visualización de camas**».



Figura E.2: Menú de administración.

Gesión de usuarios

Esta pantalla solo está disponible si el usuario tiene un rol de administrador. En esta pantalla se muestra una lista de todos los usuarios registrados en el sistema. Al mantener pulsado uno de ellos queda seleccionado y se muestran las opciones disponibles: «**Cambiar contraseña**» y «**Eliminar**».

- Si escogemos «**Cambiar contraseña**» se muestra una pantalla donde debemos introducir dos veces la nueva contraseña y pulsar el botón «Cambiar contraseña».
- Si escogemos «**Eliminar**» aparece un cuadro de diálogo preguntando al usuario si desea eliminar de forma permanente al usuario seleccionado.
- Para **añadir un nuevo usuario** al sistema debemos pulsar en el símbolo «+» en la esquina inferior derecha de la pantalla, la cual nos llevará a una pantalla donde deberemos introducir el nombre de usuario y la contraseña del nuevo usuario, y pulsar el botón «Añadir usuario».

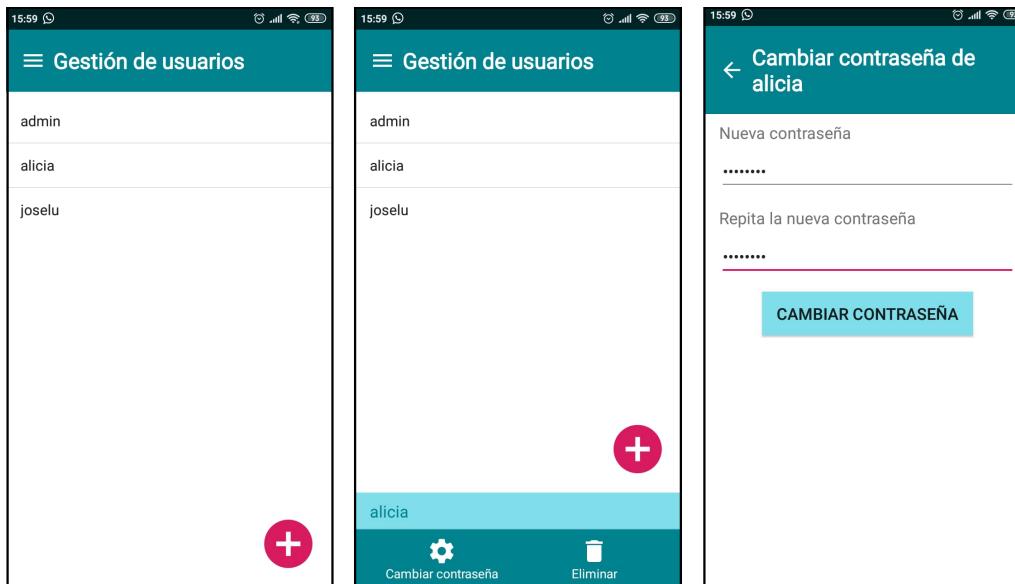


Figura E.3: Gestión de usuarios, cambiar contraseña.

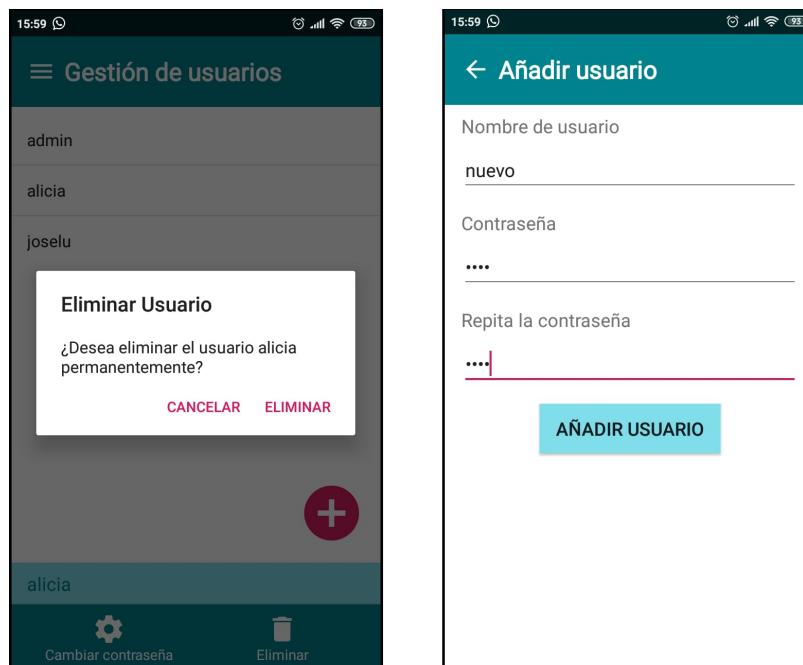


Figura E.4: Gestión de usuarios, eliminar y añadir usuarios.

Gestión de camas

Esta pantalla solo está disponible si el usuario tiene un rol de administrador. En esta pantalla se muestra una lista de todas las camas registradas en el sistema. Al mantener pulsada una de ellas queda seleccionada y se muestran las opciones disponibles: «**Modificar**», «**Eliminar**» y «**Asignar usuarios**».

De estas tres funciones, en esta versión de la aplicación solo está disponible la de **Asignar usuarios**, al seleccionar el resto solo se muestra un cuadro de diálogo informativo.² Seleccionar la opción «Asignar usuarios» lleva a una pantalla donde se muestra una lista de todos los usuarios del sistema. En esta lista se pueden marcar aquellos que se desea que puedan ver la cama seleccionada y desmarcar los que no.

Según la figura E.5 el usuario «joselu» tendría acceso a los datos de la «Cama 1» pero el usuario «alicia» no.

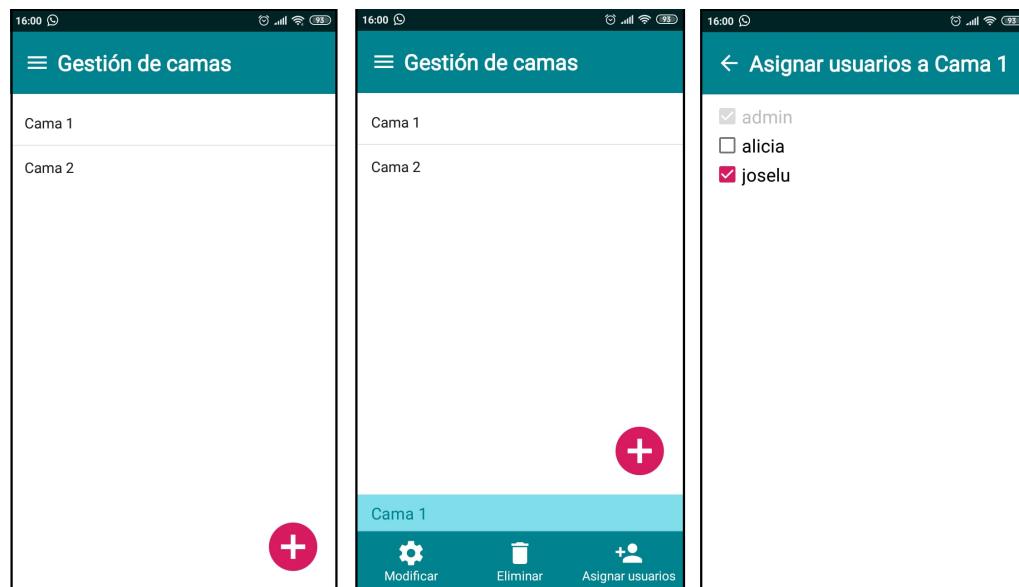


Figura E.5: Gestión de camas, asignar usuarios.

²El resto de funciones no se ha implementado debido a la forma en la que se simula la recogida de datos en tiempo real desde el servidor, no por falta de tiempo.

Visualización de camas

Esta pantalla está disponible para usuarios con un rol de administrador y con un rol de usuario genérico. Los administradores tendrán visibles todas las camas del sistema, los usuarios genéricos solo aquellas que el administrador les haya asignado.

La pantalla muestra una lista de camas junto con el estado del paciente asociado en tiempo real («dormido», «crisis epiléptica», «cama vacía» o «datos insuficientes»). Al pulsar una de las camas se accede a la pantalla de «Visualización de los datos de una cama en tiempo real».



Figura E.6: Visualización de camas.

Visualización de los datos de una cama en tiempo real

Esta pantalla está disponible para usuarios con un rol de administrador y con un rol de usuario genérico. Un usuario genérico solo podrá acceder a la pantalla asociada a una cama concreta si tiene permisos para visualizar esa cama.

Como se aprecia en la figura E.7, esta pantalla se divide en tres pestañas que contienen distintas gráficas cuyos datos se actualizan en tiempo real:

- La primera contiene una gráfica con la **probabilidad** de que el paciente esté sufriendo una crisis epiléptica.
- La segunda contiene una gráfica que muestra **una línea de datos por cada sensor de presión** instalado en el interior del colchón.

- La tercera contiene cinco gráficas relativas a las **constantes vitales** del paciente:
 - Frecuencia cardíaca (pulsaciones/minuto)
 - Frecuencia respiratoria (respiraciones/minuto)
 - Volumen sistólico (mililitros)
 - Variabilidad de la frecuencia cardíaca (milisegundos)
 - Tiempo entre pulsaciones (milisegundos)

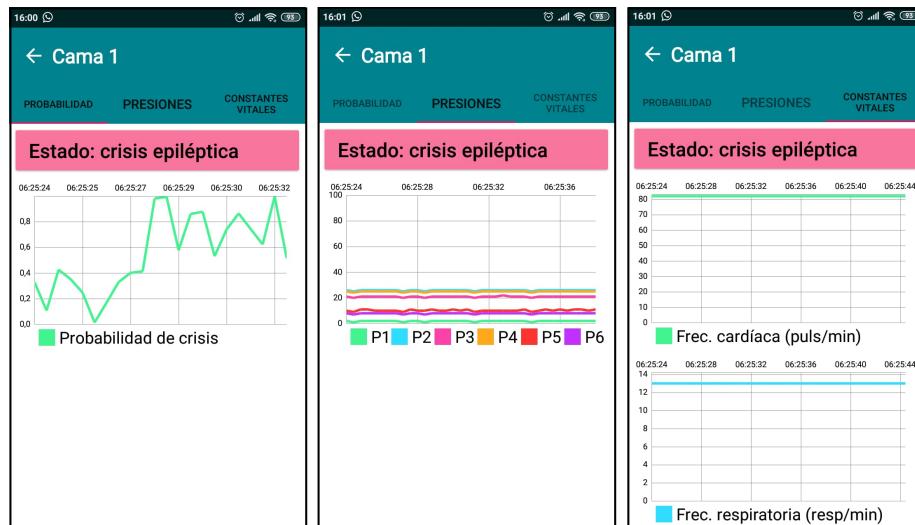


Figura E.7: Visualización de los datos de una cama en tiempo real.

Menú de navegación

En las pantallas en las que se puede ver un ícono de menú en la esquina superior izquierda es posible desplegar un menú de navegación pulsando el ícono o deslizando el dedo por la pantalla de izquierda a derecha. Como se aprecia en la figura E.8, este menú contiene opciones distintas si se trata de un usuario con rol de administrador o un usuario genérico.

Para un usuario administrador incluye las mismas opciones que para un usuario genérico y adicionalmente, las opciones del menú de administración. De esta forma se proporciona al administrador otra forma de navegar por la aplicación. Además de estas opciones se muestran tres más: «**Modificar contraseña**», «**Acerca de**» y «**Cerrar sesión**».

La opción «Cerrar sesión» devuelve al usuario a la página inicial, las otras dos opciones llevan a sus respectivas pantallas.

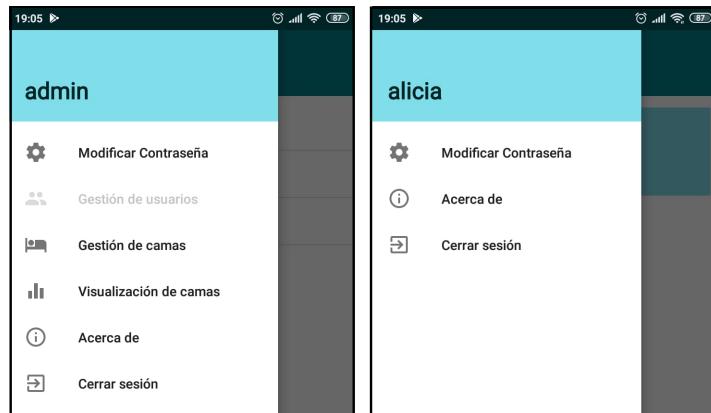


Figura E.8: Menú de navegación para un administrador y para un usuario genérico.

Modificar contraseña

Hemos visto que el administrador puede cambiar la contraseña de cualquier usuario del sistema, pero mediante la opción «Modificar contraseña» del menú de navegación se permite cambiar la contraseña del usuario que se encuentra identificado en el sistema. En esta pantalla se debe introducir una vez la antigua contraseña, dos veces la contraseña por la que se desea modificar y pulsar el botón «Cambiar contraseña».

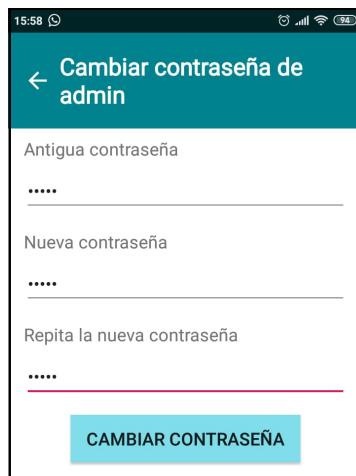


Figura E.9: Modificar contraseña.

Acerca de

Mediante la opción «Acerca de» del menú de navegación se permite acceder a una pantalla con información general sobre la aplicación, con una breve explicación de la misma, su licencia, un enlace al repositorio y la licencia de las herramientas que se han utilizado en su desarrollo.



Figura E.10: Acerca de la aplicación.

Bibliografía

- [1] Android Developers. Android developers. <https://developer.android.com/>. [Internet; descargado 17-junio-2019].
- [2] Maximilian Christ et al. tsfresh. <https://tsfresh.readthedocs.io/en/latest/>, 2019. [Internet; accedido 09-mayo-2019].
- [3] 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.
- [4] Google. Material design. <https://material.io/design/>, 2019.
- [5] Mira Cómo Hacerlo. Cómo activar los orígenes desconocidos en android. <https://miracomohacerlo.com/activar-los-origenes-desconocidos-android/>, 2017.
- [6] 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.
- [7] Scikit learn devolopers. sklearn.base.transformermixin. <https://scikit-learn.org/stable/modules/generated/sklearn.base.TransformerMixin.html>, 2018. [Internet; accedido 09-mayo-2019].
- [8] Scikit learn devolopers. sklearn.ensemble.randomforestclassifier. <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>, 2018. [Internet; accedido 09-mayo-2019].

- [9] Scikit learn devolopers. sklearn.manifold.mds. <https://scikit-learn.org/stable/modules/generated/sklearn.manifold.MDS.html>, 2018. [Internet; accedido 09-mayo-2019].
- [10] Scikit learn devolopers. sklearn.svm.oneclasssvm. <https://scikit-learn.org/stable/modules/generated/sklearn.svm.OneClassSVM.html>, 2018. [Internet; accedido 09-mayo-2019].
- [11] Overleaf. Overleaf documentation. <https://es.overleaf.com/learn>, 2019. [Internet; accedido 09-mayo-2019].
- [12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [13] tmux. tmux - home. <https://github.com/tmux/tmux/wiki>, 2019. [Internet; accedido 09-mayo-2019].
- [14] Wikipedia. Gnu afferro general public license — wikipedia, la enciclopedia libre, 2017.
- [15] Wikipedia. Modelo-vista-controlador — wikipedia, la enciclopedia libre. <https://es.wikipedia.org/w/index.php?title=Modelo%E2%80%93vista%E2%80%93controlador&oldid=116617617>, 2019.



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 – 30 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 etiquetas	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	<i>Ensembles</i> 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 tsfresh	42
9.4.	Selección del mejor conjunto de características	43
10	<i>Ensembles</i> 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.

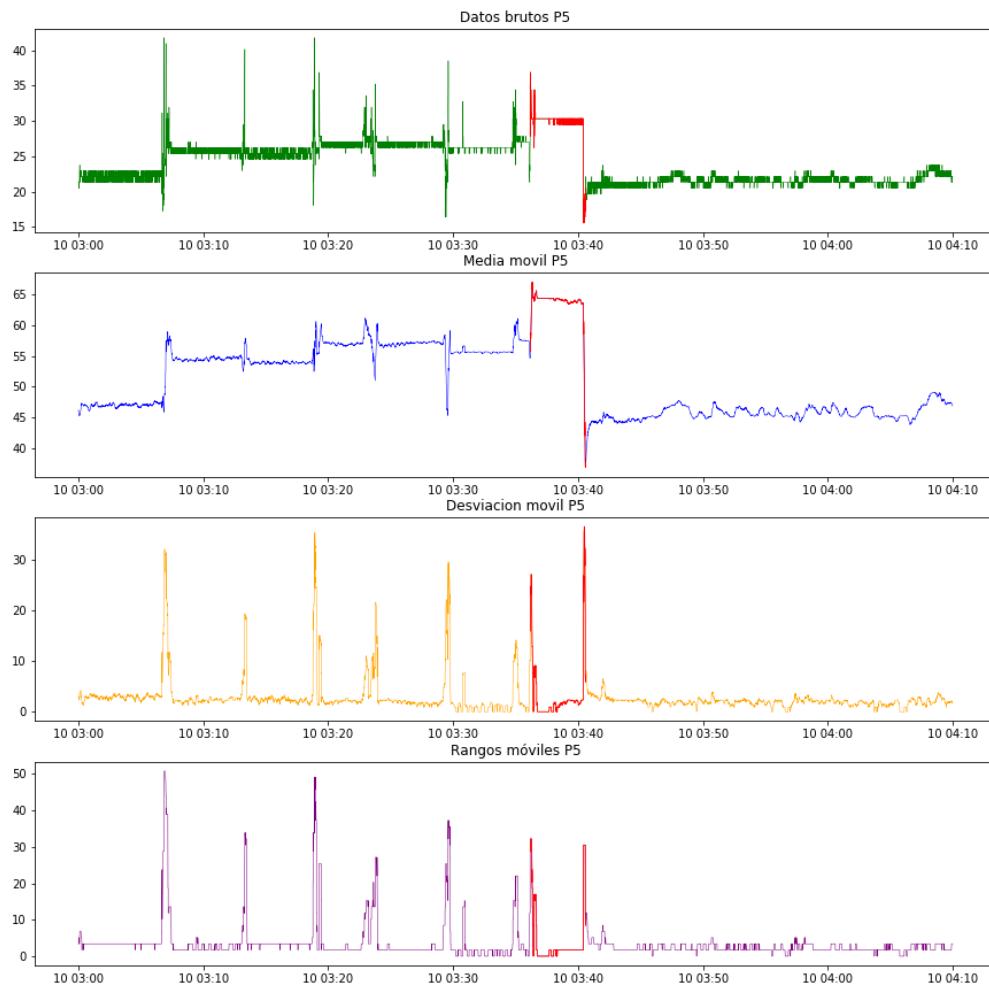


Figura 1.1: Señales de los datos brutos y estadísticos asociados al tubo de presión P5. Estimación de la crisis epiléptica en rojo.

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

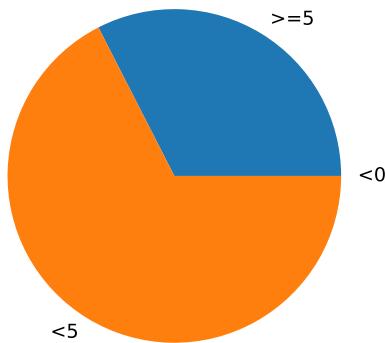
El filtrado es otra técnica empleada para eliminar el ruido, hemos empleado varias filtros como el *butterworth* [16] con valores $N=3$ y $Wn=0.05$ cuyo resultado se puede ver en la Figura 2.2a. Independientemente del tipo de filtro se aplican con la función `scipy.signal.filtfilt`. También se han probado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

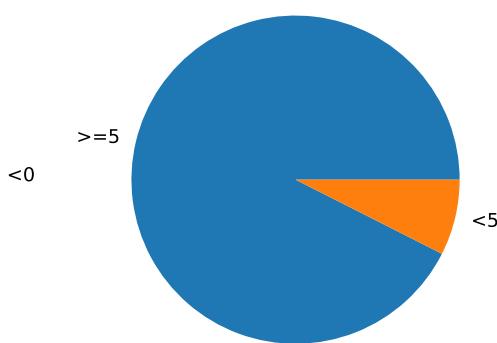
Para hacer más pequeño el conjunto de datos eliminamos aquellos tubos cuya variabilidad sea menor a un umbral, por defecto ese umbral de de 0.5.

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

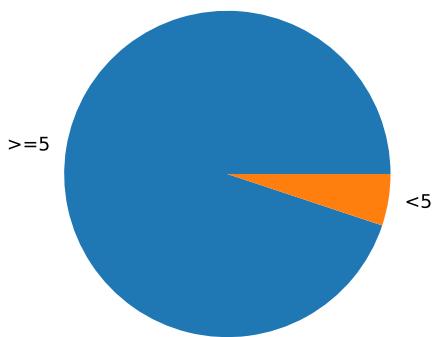
²Este ha sido el criterio para determinar al valor 5 como el mínimo para un valor real de presión



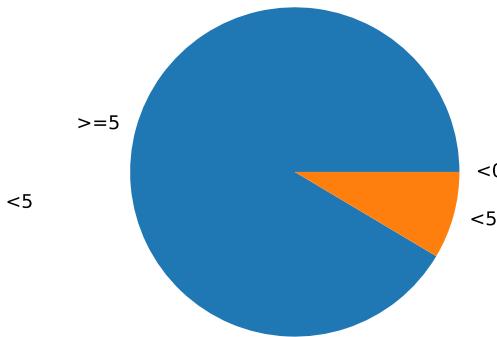
(a) Proporción de ruido en tubo de presión 1



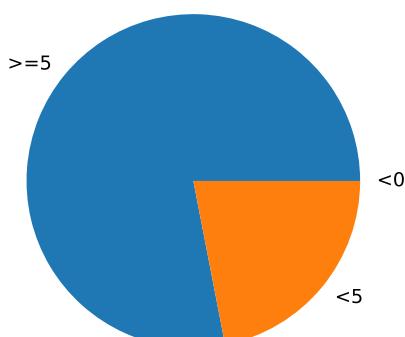
(b) Proporción de ruido en tubo de presión 2



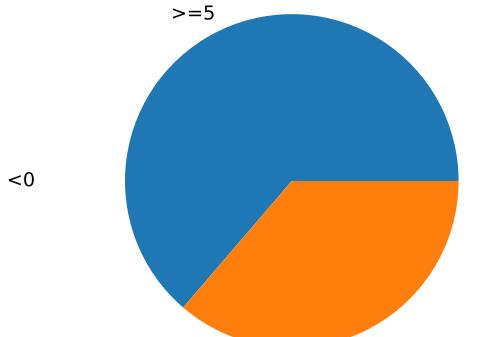
(c) Proporción de ruido en tubo de presión 3



(d) Proporción de ruido en tubo de presión 4

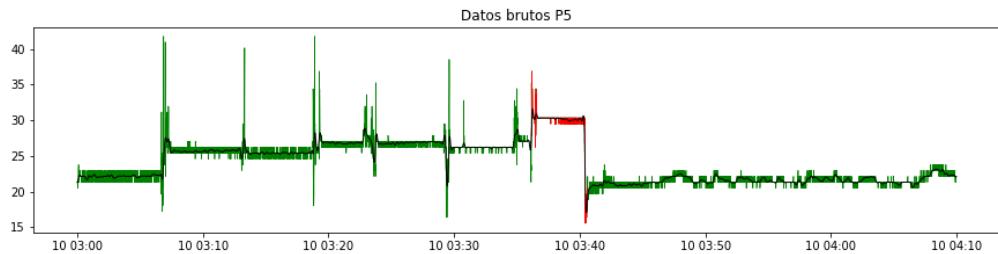


(e) Proporción de ruido en tubo de presión 5

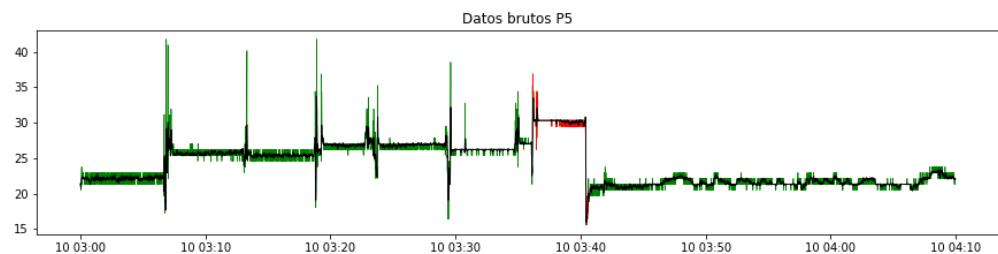


(f) Proporción de ruido en tubo de presión 6

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 etiquetas

Los transformadores de etiquetas 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

12APÍTULO 4. ANÁLISIS DE COMPONENTES PRINCIPALES (PCA)

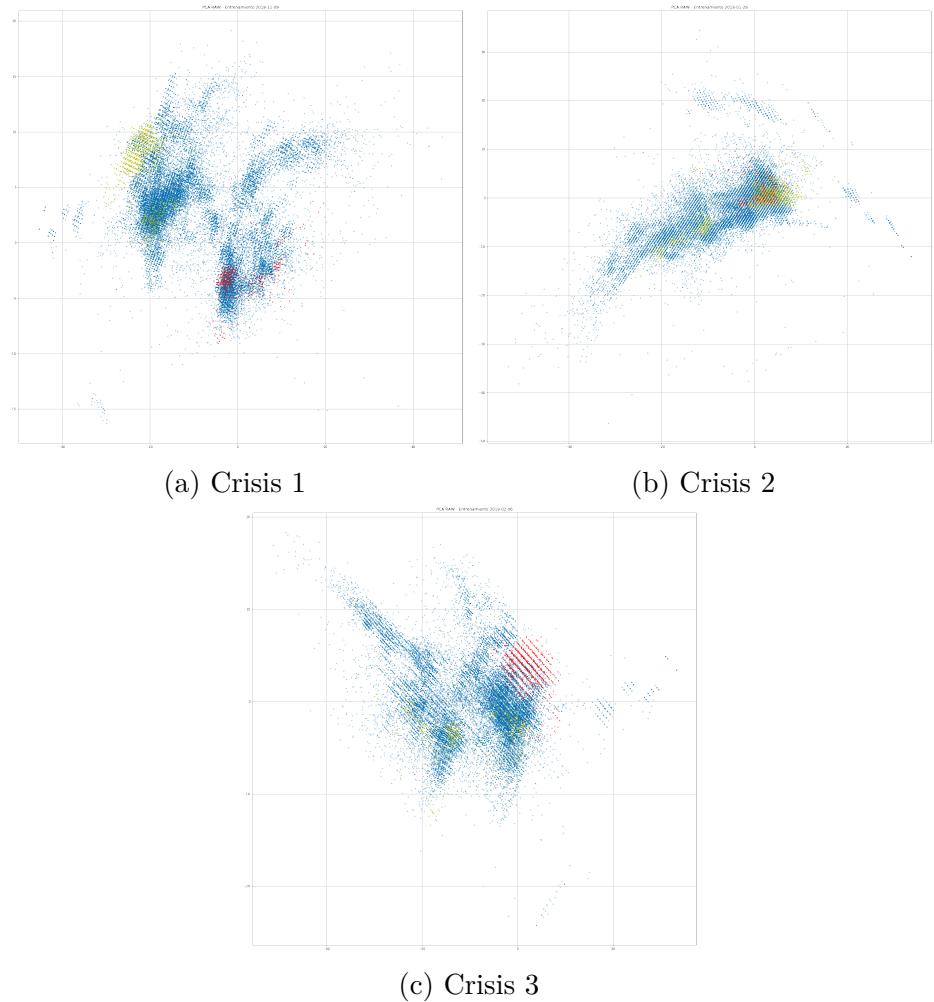


Figura 4.1: PCA ajustada a cada crisis

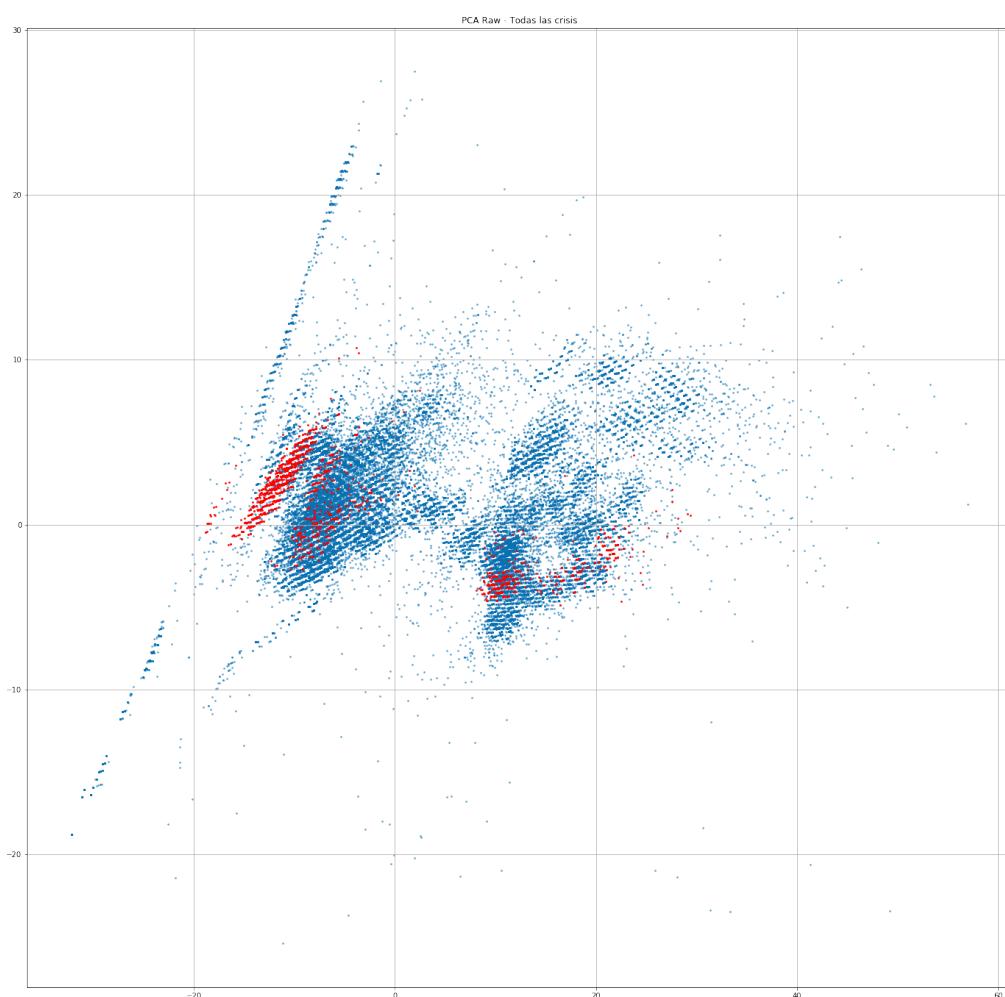


Figura 4.2: PCA ajustada con todas las crisis

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 o realizar predicciones sobre nuevos datos.

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.

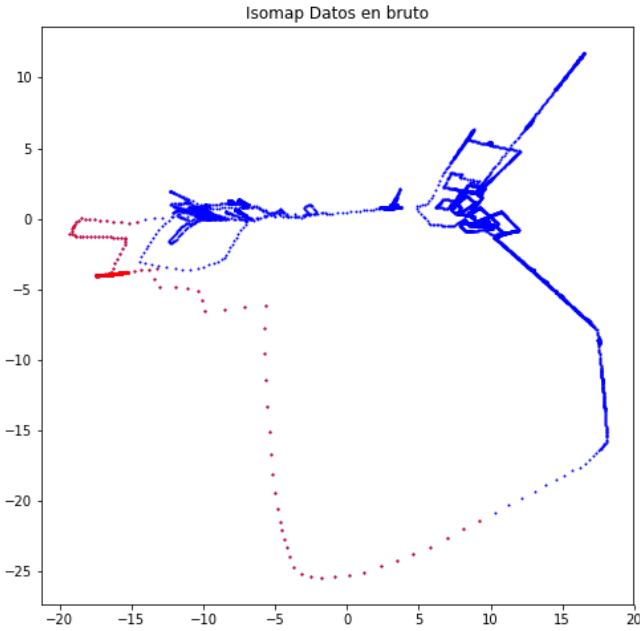


Figura 5.1: Proyección Isomap 2D de los datos brutos de presiones. Crisis epiléptica en rojo.

- 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).

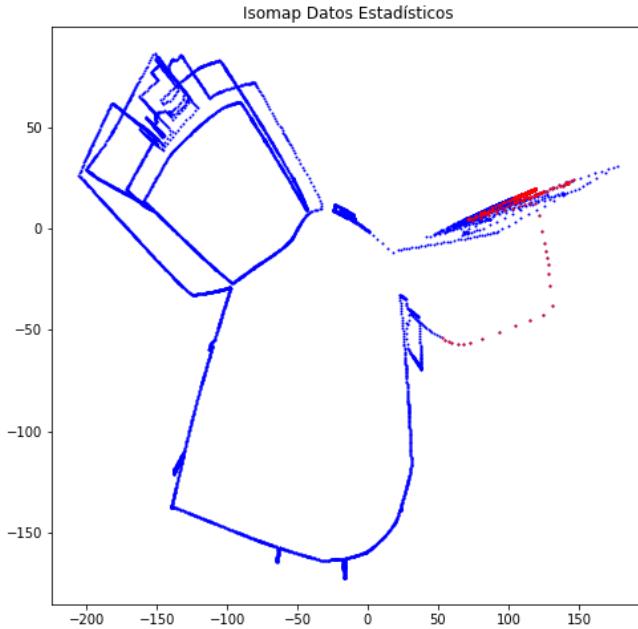


Figura 5.2: Proyección Isomap 2D de los datos estadísticos de presiones. Crisis epiléptica en rojo.

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`.

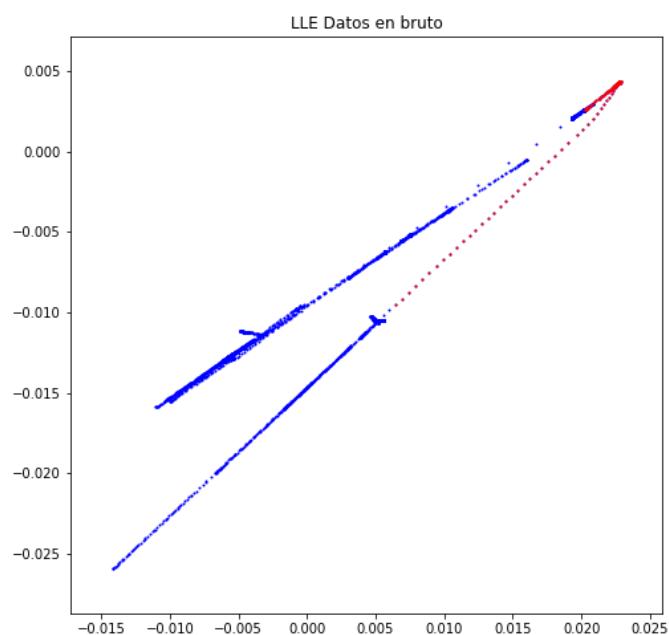


Figura 5.3: Proyección LLE 2D de los datos brutos de presiones. Crisis epiléptica en rojo.

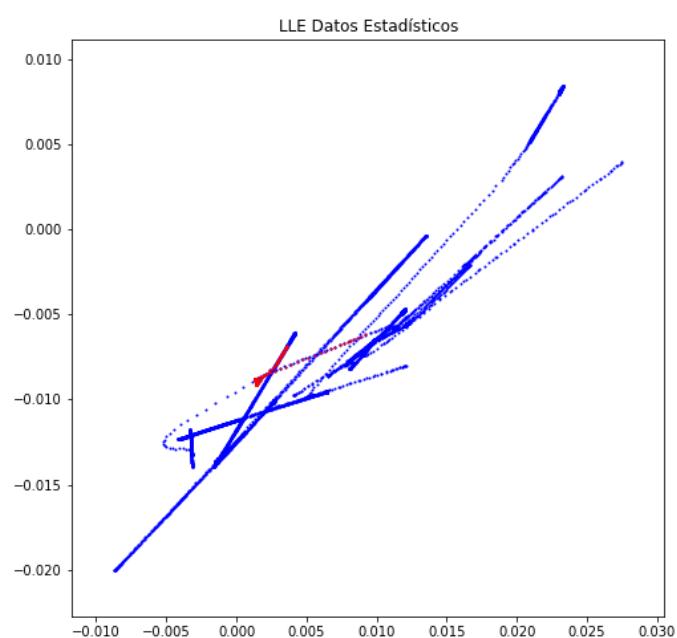


Figura 5.4: Proyección LLE 2D de los datos estadísticos de presiones. Crisis epiléptica en rojo.

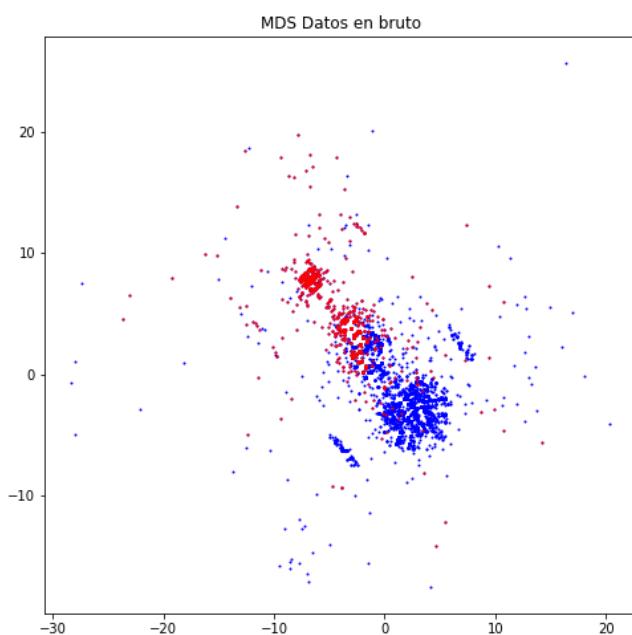


Figura 5.5: Proyección MDS 2D de los datos brutos de presiones. Crisis epiléptica en rojo.

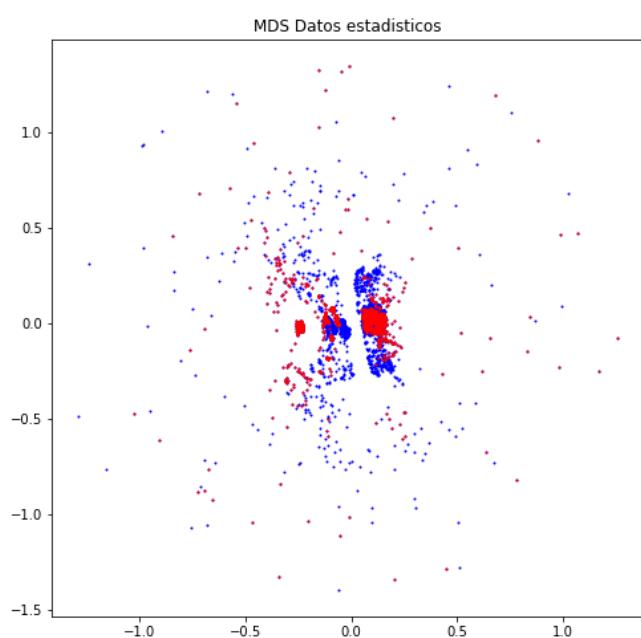


Figura 5.6: Proyección MDS 2D de los datos estadísticos de presiones. Crisis epiléptica en rojo.

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

	No Crisis	Crisis		No Crisis	Crisis
No Crisis	103893	0	No Crisis	103894	0
Crisis	535	0	Crisis	510	0

(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

	No Crisis	Crisis		No Crisis	Crisis
No Crisis	103893	0	No Crisis	103894	0
Crisis	535	0	Crisis	510	0

(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

	No Crisis	Crisis		No Crisis	Crisis
No Crisis	103893	0	No Crisis	103894	0
Crisis	535	0	Crisis	510	0

(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	9
	2	5023		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		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 (auc1^1).
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 (auc2^2).
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

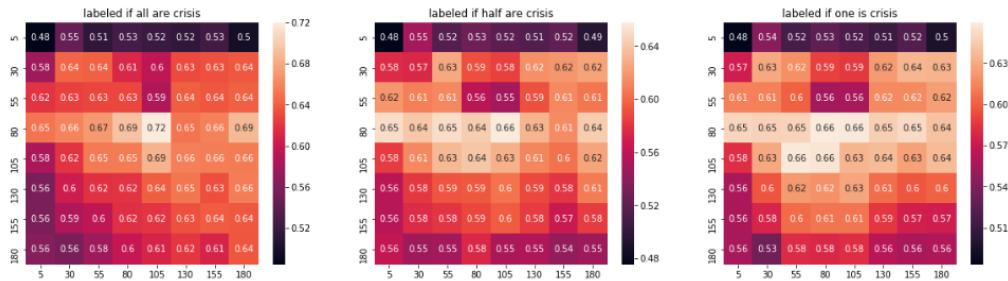


Figura 8.1: Mapa de calor de las áreas bajo la curva calculadas en la primera exploración con la métrica=ROC.

- 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

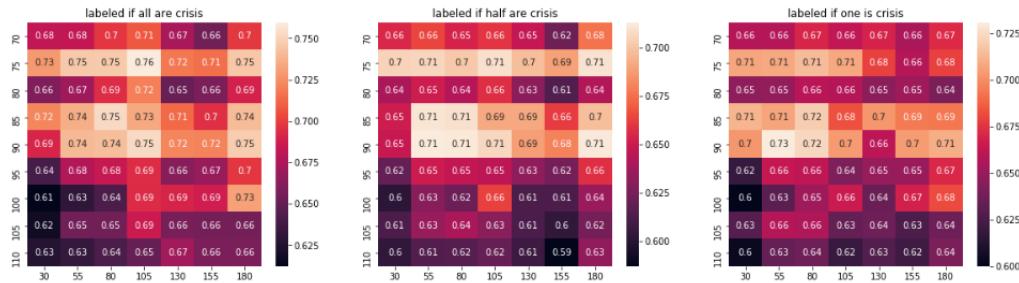


Figura 8.2: Mapa de calor de las áreas bajo la curva calculadas en la segunda exploración con la métrica=ROC.

- 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

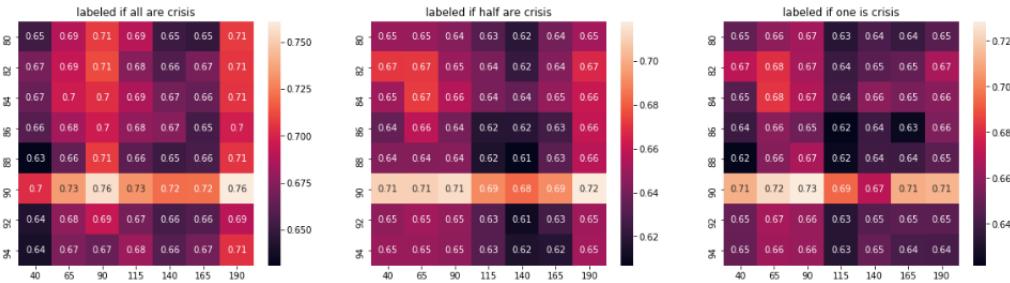


Figura 8.3: Mapa de calor de las áreas bajo la curva calculadas en la tercera exploración con la métrica=ROC.

- 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

38

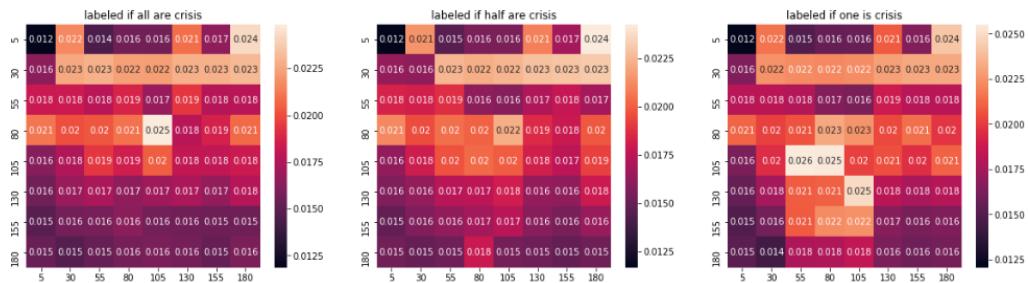


Figura 8.4: Mapa de calor de las áreas bajo la curva calculadas en la primera exploración con la métrica=Precision-Recall.

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

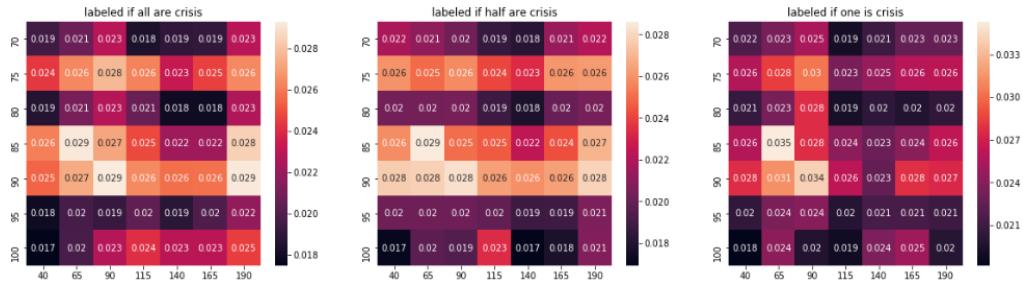


Figura 8.5: Mapa de calor de las áreas bajo la curva calculadas en la segunda exploración con la métrica=Precision-Recall.

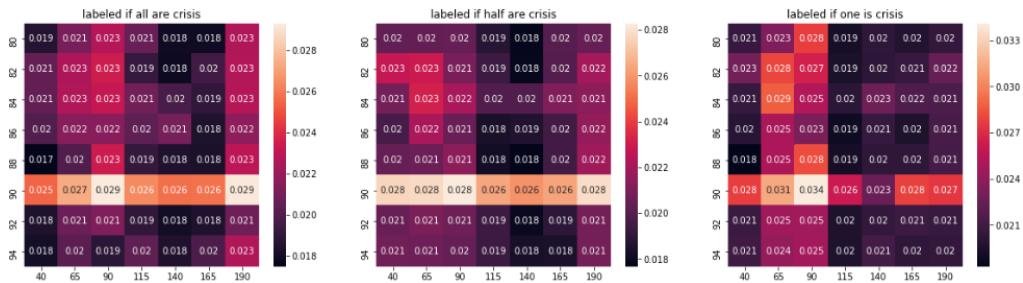


Figura 8.6: Mapa de calor de las áreas bajo la curva calculadas en la tercera exploración con la métrica=Precision-Recall.

- 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 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`.

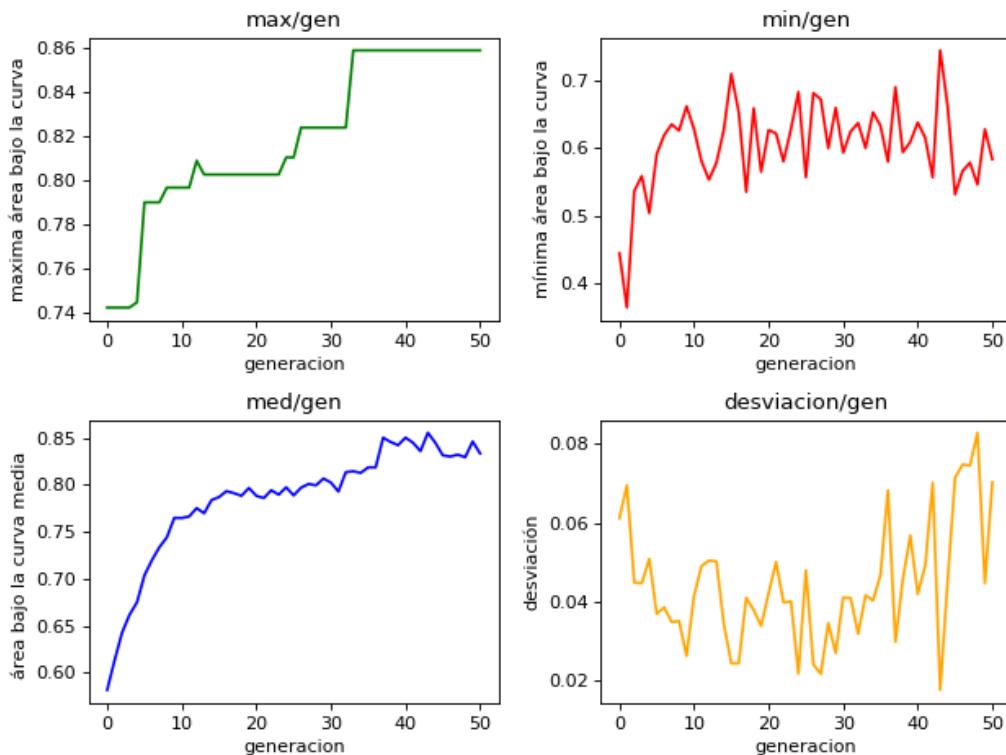


Figura 9.1: Gráficas de la evolución de las poblaciones en el algoritmo genético con la métrica=ROC.

- `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

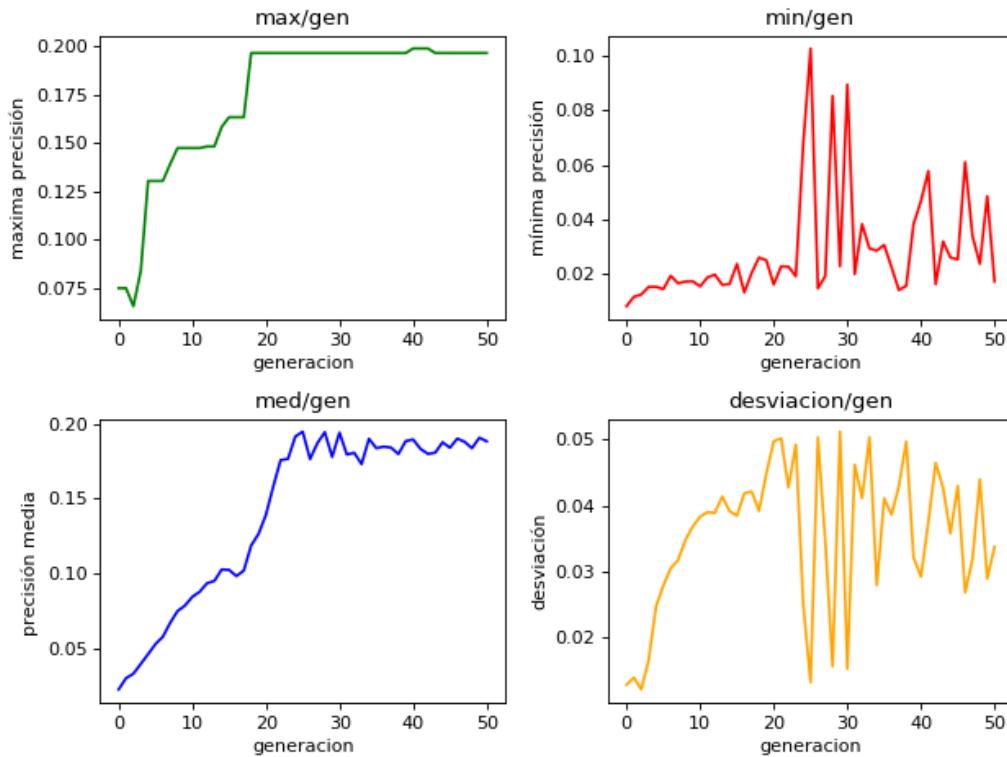


Figura 9.2: Gráficas de la evolución de las poblaciones en el algoritmo genético con la métrica=Precision-Recall.

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

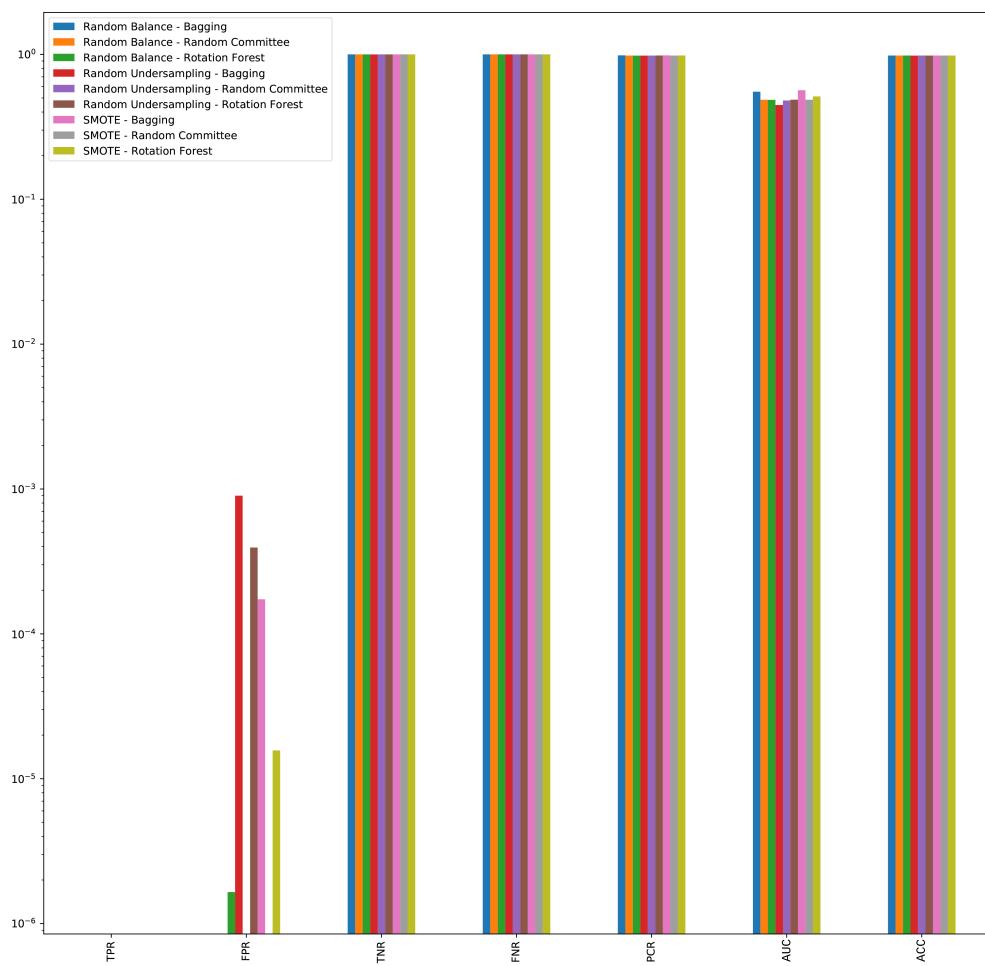


Figura 10.1: Métodos para desbalanceados - Entrenamiento con la primera crisis, testeо con la segunda crisis.

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.

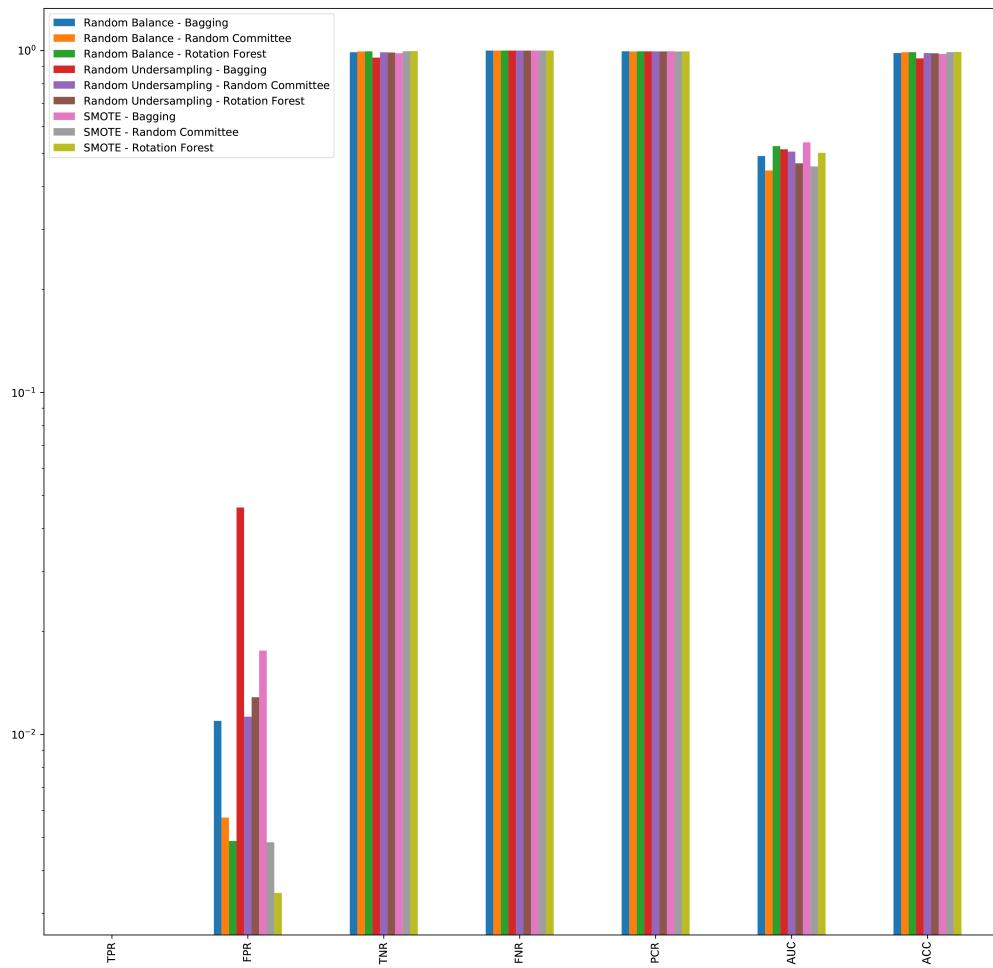


Figura 10.2: Métodos para desbalanceados - Entrenamiento con la segunda crisis, testeo con la primera crisis.

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.