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

Presentado por Alicia Olivares Gil
en Universidad de Burgos — 16 de junio de 2019
Tutores: Álvar Arnaiz González y José Francisco Díez Pastor

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. Álvar Arnaiz González, profesor del departamento de Ingeniería Civil, área de Lenguajes y Sistemas Informáticos.

Expone:

Que el alumno D. Alicia Olivares Gil, con DNI 71299943N, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado "SmartBeds - Aplicación de técnicas de minería de datos para la detección de crisis epilépticas".

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 16 de junio de 2019

Vº. Bº. del Tutor:	Vº. Bº. del co-tutor:
D. Álvar Arnáiz González	D. José Francisco Díez Pastor

Resumen

En este primer apartado se hace una **breve** presentación del tema que se aborda en el proyecto.

Descriptores

Palabras separadas por comas que identifiquen el contenido del proyecto Ej: servidor web, buscador de vuelos, android ...

Abstract

A **brief** presentation of the topic addressed in the project.

Keywords

keywords separated by commas.

Índice general

Índice general III

Índice de figuras V

Índice de tablas VI

Introducción 1
1.1. Estructura de la memoria 2
1.2. Materiales adjuntos 2

Objetivos del proyecto 5
2.1. Objetivos generales 5
2.2. Objetivos técnicos 6
2.3. Objetivos personales 7

Conceptos teóricos 9
3.1. Limpieza de datos 9
3.2. Integración de datos 12
3.3. Selección de datos 13
3.4. Transformación de datos 13
3.5. Minería de datos 18
3.6. Evaluación de patrones 21
3.7. Presentación del conocimiento 24

Técnicas y herramientas 25
4.1. Técnicas metodológicas 25
4.2. Herramientas en la fase de investigación 26

4.3. Herramientas en la fase de diseño de la aplicación 28
4.4. Herramientas en la fase de desarrollo de la aplicación 29
4.5. Otras herramientas generales 30

Aspectos relevantes del desarrollo del proyecto **31**

Trabajos relacionados **33**

Conclusiones y Líneas de trabajo futuras **35**

Bibliografía **37**

Índice de figuras

3.1. Ejemplo de señal de presión (P5) filtrada mediante un filtro de Butterworth con N=3 y Wn=0.05. En verde la señal original y en negro la señal filtrada. 11
3.2. Ejemplo de señal de presión (P5) filtrada mediante un filtro de Svatzky-Golay con *window_length*=15 y *polyorder*=2. En verde la señal original y en negro la señal filtrada 12
3.3. Abstracción de la reducción de la dimensionalidad de los datos . 14
3.4. *PCA* aplicado a la noche de la crisis 1 (izquierda) y a la noche de la crisis 2 (derecha). 15
3.5. Comparativa de las técnicas de transformación no lineal. 16
3.6. Disposición de la matriz de confusión. 22
3.7. ROC. De izquierda a derecha: mejor caso, caso intermedio y peor caso. 23
3.8. Ejemplo de representación de la probabilidad de crisis en función del tiempo. 24

Índice de tablas

Introducción

La epilepsia es un trastorno neurológico provocado por la alteración de la actividad normal de una región cerebral, que desencadena crisis caracterizadas por convulsiones musculares reiteradas y, en ocasiones, pérdida de la consciencia. Se trata de una de las enfermedades neurológicas más habituales, y aunque las crisis epilépticos pueden experimentarse de forma aislada o durante periodos de tiempo limitados, una gran cantidad de la población las sufre de forma crónica. Según la Federación Española de Epilepsia [2] alrededor de 700 000 personas en España padecen o han padecido epilepsia a lo largo de su vida, y más de 200 000 la padecen de forma activa.

Para una persona con epilepsia crónica, la detección inmediata de una crisis es vital para permitir la aplicación de primeros auxilios que ayuden a evitar consecuencias permanentes. Actualmente, en la bibliografía se habla de varias técnicas para la detección automática de crisis [6, 10], la mayoría basadas en Electroencefalogramas (EEG) o en el uso de dispositivos portátiles (*wearables*) como pulseras inteligentes basadas en la monitorización de las constantes vitales del paciente.

La detección mediante EEG se usa principalmente para diagnósticos médicos, ya que los dispositivos que se necesitan son demasiado costosos o aparatosos como para ser usados en el día a día del paciente. Por otro lado, las pulseras inteligentes resultan más convenientes para este cometido, ya que son más baratas y cómodas de utilizar. Sin embargo, ambas técnicas requieren del uso consciente y continuado de los dispositivos de detección, lo que puede suponer un inconveniente para pacientes dependientes o con necesidades especiales. Por esta razón se propone el uso de colchones inteligentes para la detección automática de crisis nocturnas, mediante sensores de presión y biométricos incorporados en el interior del propio colchón.

Sea cual sea el dispositivo utilizado, la captación de los datos (actividad eléctrica del cerebro, constantes vitales, presiones, etc.) no basta para detectar una crisis. Es necesario un procesado adecuado para determinar si estos datos corresponden o no con una crisis epiléptica. Para ello, las técnicas de minería de datos permiten generar modelos de clasificación capaces de realizar esta tarea. Para este trabajo de fin de grado, el principal objetivo será encontrar un modelo de clasificación efectivo mediante la aplicación de este tipo de técnicas sobre los datos disponibles.

1.1. Estructura de la memoria

Esta memoria incluye los siguientes apartados:

- **Objetivos del proyecto:** se definen los objetivos generales, técnicos y personales que se persiguen con la realización de este trabajo.
- **Conceptos teóricos:** TODO
- **Técnicas y herramientas:** TODO
- **Aspectos relevantes del desarrollo del proyecto:** TODO
- **Trabajo relacionados:** TODO
- **Conclusiones y líneas de trabajo futuras:** TODO

1.2. Materiales adjuntos

- **Anexos:**
  - Plan de Proyecto Software
  - Especificación de Requisitos
  - Especificación de diseño
  - Documentación técnica de programación
  - Documentación de usuario
- **Cuaderno de investigación:** recoge las explicaciones de todas las técnicas probadas, los resultados y las comparativas de todos los experimentos realizados con el fin de encontrar el mejor modelo de clasificación para el problema.

- **Experimentos:** Conjunto de notebooks de jupyter que contienen todos los experimentos realizados.
- **App de Android:** Archivo .apk para la distribución e instalación de la aplicación para Android desarrollada con el fin de mostrar la aplicabilidad del modelo de clasificación.

A la memoria y a todos los demás materiales adjuntos se puede acceder a través del repositorio del proyecto en GitHub: https://github.com/aog0036/TFG-SmartBeds.

Objetivos del proyecto

En este apartado se exponen los objetivos perseguidos con la realización de este trabajo:

2.1. Objetivos generales

- Investigar sobre técnicas del estado del arte aplicadas a problemas similares.
- Aplicar técnicas de minería de datos siguiendo los pasos del Descubrimiento de Conocimiento en Bases de Datos (*KDD*) [4].
- Explorar, aplicar y comparar distintas formas de preprocesado de los datos (filtrado, normalización, transformación, etc.).
- Usar técnicas de proyección de datos a dos dimensiones para comprobar si los instancias de «crisis» son fácilmente separables de las instancias de «no crisis».
- Probar modelos de clasificación para conjuntos de datos con preprocesados basados en estadísticas simples.
- Probar modelos de clasificación para conjuntos de datos con preprocesados basados en características de series temporales.
- Probar modelos de clasificación mediante *ensembles* para conjuntos de datos desequilibrados.
- Probar detección de anomalías mediante un modelo *One-Class*.

- Comparar el rendimiento de los modelos obtenidos.
- Comparar distintas métricas usadas para evaluar el rendimiento los modelos obtenidos.
- Generar un modelo de clasificación capaz de detectar crisis epilépticas a partir de los datos disponibles.
- Desarrollar una app de Android para mostrar la aplicabilidad del modelo de clasificación generado.

2.2. Objetivos técnicos

- Aprender LaTeX, en concreto su uso en herramientas de escritorio y online (tales como Overleaf).
- Realizar y exponer los resultados de los experimentos en notebooks de jupyter empleando código Python.
- Usar bibliotecas de minería de datos en Python, en concreto sklearn [5], para la proyección de los datos y para la obtención y evaluación de los modelos.
- Generar un conjunto de Transformadores compatibles con sklearn para poder aplicar las técnicas de preprocesado de forma directa.
- Utilizar bibliotecas de procesamiento y visualización de datos tales como Pandas y Matplotlib.
- Usar la biblioteca tsfresh [1] para extracción de características en series temporales.
- Usar la herramienta Weka [3] para probar modelos de clasificación mediante ensembles en conjuntos de datos desequilibrados.
- Desarrollar una app Android con soporte para API 23 y superiores.
- Usar peticiones HTTP desde la app para la comunicación con la API del servidor remoto.
- Usar Socket.IO desde la app para la obtención de datos en tiempo real del servidor remoto.
- Usar la plataforma GitHub junto con la extensión ZenHub para la gestión del proyecto.

- Aplicar la metodología Scrum adaptada a un proyecto con fines educativos.
- Realizar test TODO.

2.3. Objetivos personales

- Iniciarme en el campo de la investigación.
- Explorar técnicas y herramientas aplicadas a la minería de datos.
- Aprender a generar documentación con LaTeX.
- Iniciarme en el desarrollo de aplicaciones Android.

Conceptos teóricos

Este proyecto es principalmente de investigación, por lo que tiene una importante carga teórica. En el apéndice del cuaderno de investigación se explica el uso de todas las técnicas que hemos empleado en los experimentos, especificando implementaciones concretas, parámetros utilizados y resultados obtenidos. Por otra parte, en este apartado se van a exponer los conceptos teóricos en los que se basan esas técnicas sin entrar en los detalles de implementación.

Dado que el desarrollo de la investigación se ha basado en el proceso *KDD* (explicado más detenidamente en el apartado de Técnicas y Herramientas), este apartado se estructurará en función de los pasos que incluye este proceso:

3.1. Limpieza de datos

Los datos de los que disponemos tienen los siguientes atributos:

- **MAC_NGMATT** y **UUID_BSN**: identificador del colchón asociado alos tubos de presión e identificador del sensor de biométricas respectivamente.
- **DateTime**: día y hora en la que fue tomada cada instancia de dato.
- **P1, P2, P3, P4, P5, P6, P7, P8, P9, P10, P11, P12**: presiones, en mBar, captadas por los 12 tubos de presión integrados en el colchón.
- **HR, RR, SV, HRV, B2B**: valores de las constantes vitales que corresponden con la frecuencia cardíaca (pulsaciones/minuto), la frecuencia respiratoria (respiraciones/minuto), el volumen sistólico (mili-

litros), la variabilidad del ritmo cardíaco (milisegundos) y el tiempo entre pulsaciones (milisegundos) respectivamente.

- **SS**: potencia de la señal relativa a la matriz de tubos de presión. Un valor superior a 400 se considera aceptable.

- **STATUS**: estado de medición del sensor de constantes vitales. 0=*low signal*, 1=*ok signal*, 2=*high signal*, 3= *close to overload*, 4= *close to max HR*.

Por otra parte se nos proporcionaron unos rangos aproximados de horas en los que el paciente tuvo un ataque epiléptico.

Eliminación de instancias por baja señal

Dos de estos atributos, SS y STATUS, hacen referencia a la calidad de la instancia de los datos, uno haciendo referencia a la matriz de tubos de presión y otro al sensor de constantes vitales, y dado que la fase de limpieza de datos consiste en eliminar el ruido y las instancias incorrectas, es aquí donde eliminaremos todas las instancias cuyos datos no consideremos fiables en función de los valores de estos dos atributos.

Concretamente eliminamos todas aquellas instancias que tengan un SS menor de 400, pero no tendremos en cuenta el valor de STATUS ya que, como se explicará más adelante, los valores de las constantes vitales no se han usado para la búsqueda del modelo de clasificación.

Eliminación manual del ruido

Una opción para eliminar el ruido de los datos es hacerlo de forma manual, inspeccionando los datos disponibles y modificar o eliminar aquellos que sean inconsistentes.

Mediante una inspección inicial de los datos se detectó que algunos datos de los tubos de presión en ocasiones eran negativos (lo cual no debería ocurrir), y aparecían valores bajos de presión en momentos en los que la cama debería estar vacía. Por esta razón, se decidió considerar todo valor de presión menor a 5 como ruido convirtiéndolo a 0 (de esta forma nos deshacemos también de los valores negativos).

Filtrado

Otra forma de eliminar el ruido de una señal consiste en usar filtros. Un filtro es un sistema que se aplica a una señal (en nuestro caso la señal de presiones en el tiempo) y la transforma para conseguir un objetivo concreto, en este caso el suavizado de la señal para eliminar el ruido. Se han probado dos tipos de filtro distintos:

- **Filtro de Butterworth** [12]: Diseñado para filtrado de señales eléctricas, trata de producir la respuesta más plana que sea posible hasta la frecuencia de corte (Wn), dicho de otra forma, trata de mantener intactas las frecuencias por debajo de la frecuencia de corte mientras disminuye las frecuencias superiores con razón proporcional a N, siendo N el orden de filtrado:

$$|H| = \frac{1}{\sqrt{1 + (W/Wn)^{2N}}}$$

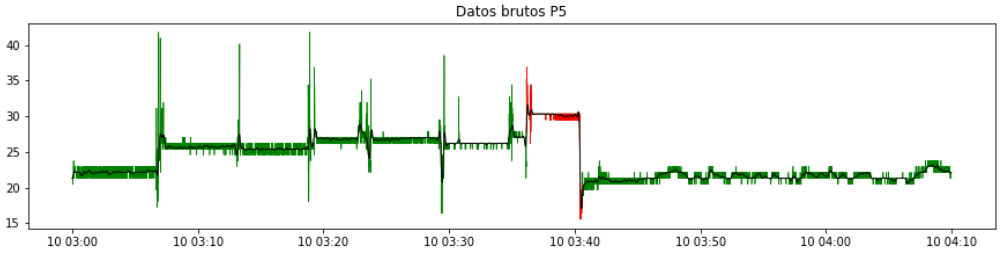

Figura 3.1: Ejemplo de señal de presión (P5) filtrada mediante un filtro de Butterworth con N=3 y Wn=0.05. En verde la señal original y en negro la señal filtrada.

- **Filtro de Savitzky–Golay** [11]: Se basa en el cálculo de una regresión polinomial local para la que se debe especificar un tamaño de ventana (*window_lenght*) y el orden del polinomio utilizado en la regresión (*polyorder*).

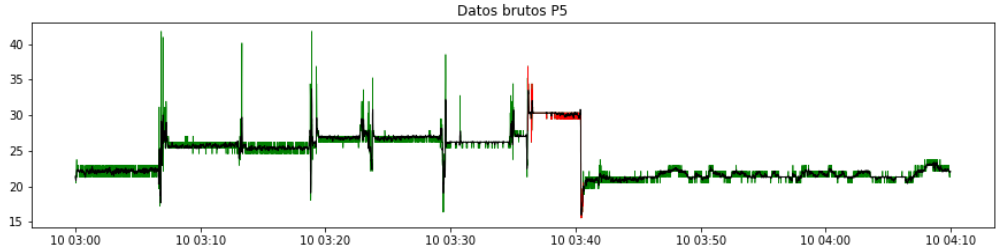

Figura 3.2: Ejemplo de señal de presión (P5) filtrada mediante un filtro de Svatzky-Golay con *window_length*=15 y *polyorder*=2. En verde la señal original y en negro la señal filtrada

Eliminación de atributos inconsistentes

En la primera inspección de los datos también se advirtió que los atributos referentes a las constantes vitales eran nulos de forma intermitente y en una gran cantidad de las instancias. Esto se debió al mal funcionamiento del sensor durante las primeras etapas de recogida de los datos.

Aunque muchas técnicas de minería de datos soportan la presencia de atributos desconocidos (*missing*) en algunas de las instancias, la cantidad de instancias con constantes vitales nulas o incorrectas era tan grande que se tomó la decisión de no tener en cuenta estos datos para el resto de experimentos.

Cabe destacar que esto supone un gran inconveniente, ya que la mayoría de técnicas de detección automática encontrados en el estado del arte (quitando las centradas en EEG) se basan en datos biométricos.

3.2. Integración de datos

Los datos disponibles corresponden con las instancias recogidas al lo largo de varios meses de una sola cama (un solo paciente). Estos datos nos han sido proporcionados en varios archivos de extensión .csv, por lo que la fase de integración ha consistido en recoger los datos en varios subconjuntos, uno por cada noche, considerando que una noche corresponde con el periodo de tiempo en el que el paciente estuvo acostado en la cama de forma ininterrumpida.

3.3. Selección de datos

La selección de los datos consiste en desechar aquellos atributos o instancias que no resultan útiles para la extracción de conocimiento. A estas alturas del proceso seguimos teniendo datos relativos a la identificación del colchón, a la hora en la que se tomó la instancia de dato y a las presiones.

Eliminación de identificadores

Al contar con los datos de un único paciente, los atributos de identificación de la cama MAC_NGMATT y UUID_BSN no proporcionan ningún tipo de información útil para la extracción de conocimiento, por lo que ni siquiera son tenidos en cuenta en la fase de integración.

Eliminación de atributos de baja variabilidad

Este proceso consiste en detectar aquellos atributos que varían menos de un umbral determinado durante el tiempo, y por lo tanto no proporcionarán información relevante.

Tras llevar a cabo este proceso se eliminaron los atributos P7, P8, P9, P10, P11 y P12. Esto tiene sentido ya que esos campos corresponden con la matriz de tubos de presión de una de las mitades del modelo de colchón de matrimonio, y el colchón con el que se trabaja en este proyecto es individual.

En este punto ya hemos seleccionado los atributos que serán considerados como **Datos en Bruto**:

- DateTime
- P1, P2, P3, P4, P5 y P6

3.4. Transformación de datos

Técnicas de reducción de la dimensionalidad

El primer tipo de transformación de datos que se utilizó fueron las técnicas de reducción de dimensionalidad. Estas técnicas consisten en proyectar datos en un plano, reduciendo su dimensión (número de atributos de cada instancia, en nuestro caso seis) a dos. Este tipo de enfoques se basan en la hipótesis de que la dimensionalidad de los conjuntos de datos es artificialmente alta,

y que se puede mantener la misma información con una representación de los mismos con una dimensionalidad menor.

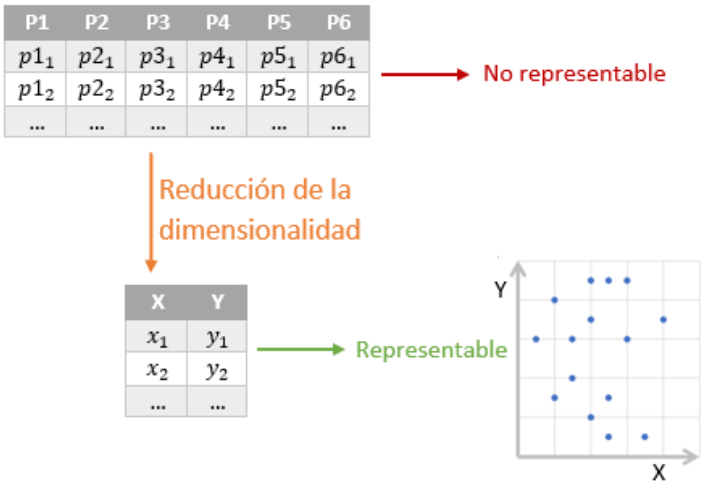

Figura 3.3: Abstracción de la reducción de la dimensionalidad de los datos

Con esto, el objetivo que se persiguió fue comprobar si las instancias de «crisis» eran fácilmente separables de las instancias de «no crisis» en una representación bidimensional de los datos. Existen varios tipos de técnicas, cada una basada en unos criterios concretos para realizar la proyección, y se pueden clasificar principalmente en transformaciones lineales y transformaciones no lineales.

Transformaciones lineales

Este tipo de transformaciones supone que los datos observados son combinación lineal de una cierta base, y en ellas se engloba el **Análisis de Componentes Principales** (*PCA*) [9].

PCA es una de las técnicas más utilizadas y consiste en escoger el nuevo sistema de coordenadas (o base) de forma que se maximice la varianza entre las instancias. Cabe destacar que para el nuevo sistema de coordenadas se puede escoger cualquier dimensionalidad menor o igual a la del conjunto original de datos, pero a mayor dimensionalidad, mayor coste de computación, por lo que solo se ha probado la proyección a dos dimensiones.

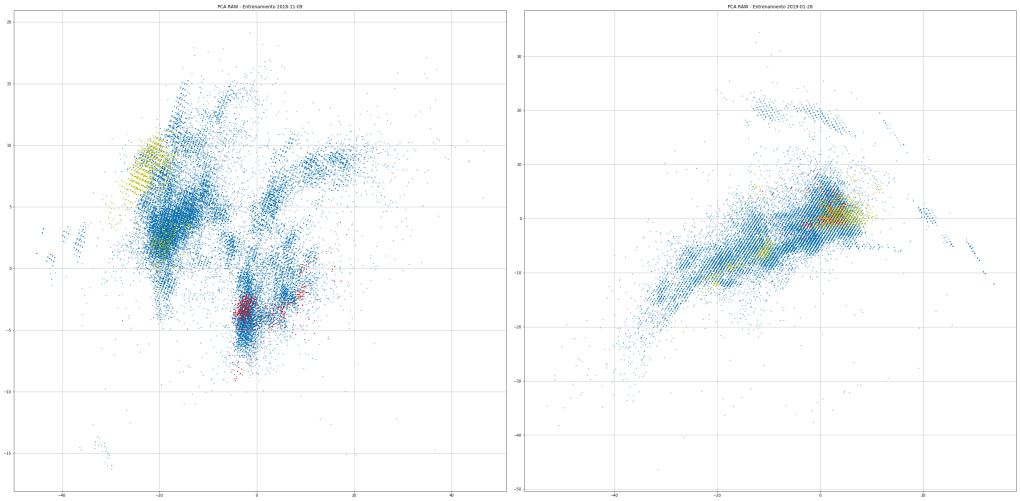

Figura 3.4: *PCA* aplicado a la noche de la crisis 1 (izquierda) y a la noche de la crisis 2 (derecha).

En la figura 3.4 los puntos rojos corresponden con instancias de «crisis» y los azules con instancias de «no crisis». Como se puede ver no existe separación entre ellas.

Transformaciones no lineales

Las transformaciones no lineales [8] no asumen que los datos sean una combinación lineal de una base. Existen varias técnicas de este tipo y en nuestro caso se han probado con éxito las siguientes:

- ***Isomap*:** Trata de reducir la dimensionalidad manteniendo las distancias geodésicas entre todas las instancias.
- ***Locally Linear Embedding* (*LLE*):** Reduce la dimensionalidad preservando las distancias dentro de los "vecindarios" locales. Se puede considerar como un conjunto de *PCA*s locales que se comparan globalmente para encontrar la mejor reducción no lineal.
- ***Multi-Dimensional Scaling* (*MDS*):** Es una técnica bastante utilizada en marketing y ciencias sociales que se basa en la similitud o disimilitud de los datos. Busca reducir la dimensionalidad tratando las distancias como valores proporcionales a la disimilitud de las instancias (lo que se parecen entre ellas).

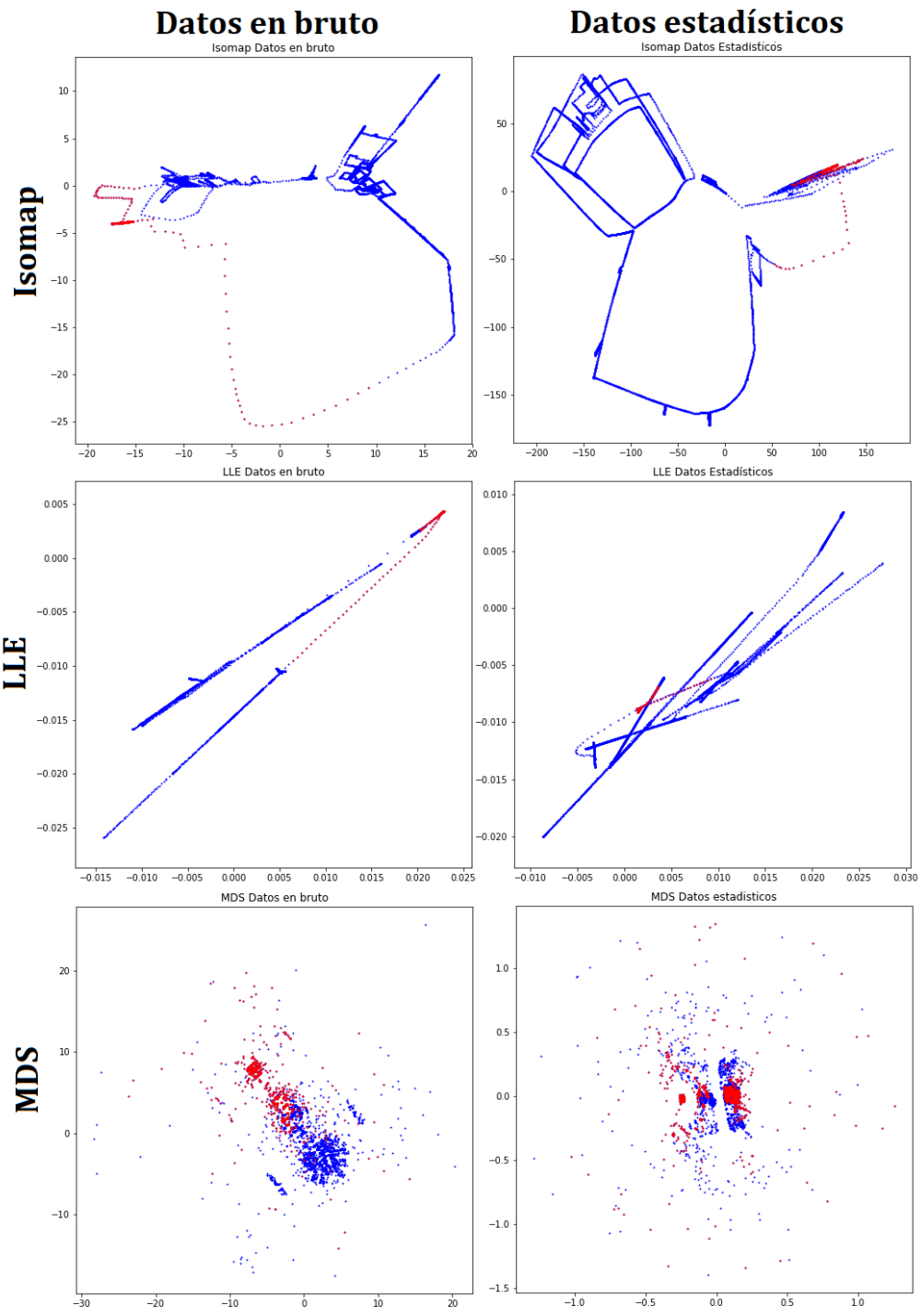

Figura 3.5: Comparativa de las técnicas de transformación no lineal.

Como se aprecia en la figura 3.5 con algunas de estas técnicas se consigue un grado algo mayor de separación, pero las instancias siguen sin ser fácilmente separables. En este caso y como se puede observar, hemos aplicado las técnicas no solo a los datos en bruto sino también a los datos estadísticos, lo que nos lleva al siguiente tipo de transformación.

Cálculo de estadísticas móviles

Las técnicas de minería de datos se pueden aplicar directamente a los datos en bruto, pero en ocasiones ofrecen un mejor rendimiento aplicadas a datos estadísticos calculados a partir de estos. Para calcular estadísticas móviles se debe definir el tamaño de la ventana (N), de forma que para cada ventana se calculará un valor estadístico a partir de los datos contenidos en ella. Se han usado tres estadísticas simples:

- **Media móvil:** Es el valor promedio de los valores contenidos en la ventana.

$$\overline{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

- **Desviación típica móvil:** Es una medida de la dispersión de los valores contenidos en la ventana.

$$\sigma = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (x_i - \overline{x})^2}$$

- **Rango móvil:** Es la diferencia entre el valor máximo y el valor mínimo de los contenidos en la ventana.

Cálculo de características de series temporales

Además de las descritas en el apartado anterior, se pueden calcular multitud de estadísticas más complejas, como es el caso de las características de series temporales.

Una serie temporal [13] no es más que una secuencia de instancias medidas en determinado momento de tiempo y ordenadas cronológicamente por lo que nuestro conjunto de datos corresponde con la definición de serie temporal. Las series temporales se suelen usar para estudiar la relación causal entre diversas variables que cambian en el tiempo y pueden influirse entre sí, y para este cometido existen una serie de técnicas específicas basadas en cálculos estadísticos más complejos.

Para este proyecto se ha empleado una biblioteca que extrae algunas de esas características a partir de una serie temporal, y hemos tratado de usarlas para generar el modelo de clasificación.

3.5. Minería de datos

En esta fase se ha tratado de generar un modelo capaz de encontrar los patrones que definen una crisis epiléptica. Para ello el modelo debe ser capaz de aprender a partir de los datos, y en función de como lo haga se puede hablar de dos tipos básicos de aprendizaje: supervisado y no supervisado.

- Un modelo de **aprendizaje supervisado** se entrena con un conjunto de datos etiquetado, es decir, que cada instancia de dato de entrenamiento tiene información sobre a qué clase pertenece (para un problema de clasificación como el nuestro), en este caso, las dos clases posibles son «crisis» o «no crisis». Tras el entrenamiento, el modelo deberá ser capaz de etiquetar nuevas instancias de dato distintas a las que se han usado en el entrenamiento como instancias de «crisis» o «no crisis» basándose en lo aprendido. En definitiva, el ciclo de vida de un clasificador supervisado tiene dos fases:
  - **Entrenamiento:** El modelo aprende a partir de un conjunto de datos etiquetado.
  - **Predicción:** El modelo predice la clase de un dato no etiquetado, es decir, cuya clase real se desconoce.
- En el **aprendizaje no supervisado** el modelo no tiene conocimiento sobre a qué clase pertenece cada instancia. Con este tipo de aprendizaje se pueden, por ejemplo, agrupar los datos en subconjuntos en función de sus características. Las técnicas de reducción de la dimensionalidad podrían ser consideradas técnicas de aprendizaje no supervisado, ya que elaboran la proyección sin conocer la clase de cada instancia, pero podrían darnos información sobre la existencia de dos clases si en las proyecciones bidimensionales existiera una clara separación entre dos clusters de instancias.

Por otra parte, dentro de los modelos de clasificación podemos distinguir entre clasificadores simples o *ensembles*.

- Los **clasificadores simples** emplean un solo modelo que será el encargado de predecir la clase de las nuevas instancias.

- Los ***ensembles*** son conjuntos de clasificadores simples cuyas predicciones se combinan para obtener una predicción final. La idea es construir varios modelos que no necesariamente predigan la misma clase para la misma instancia, y será la combinación de estas predicciones, por ejemplo, mediante votación, la que determinará la clase que predice el *ensemble*. Para que este sistema de clasificación tenga sentido los modelos que forman el *ensemble* deben ser algo distintos entre sí, de lo contrario no ofrecerían ninguna ventaja, y esa diferencia puede radicar en varios mecanismos, por ejemplo:

  - El proceso de construcción del modelo no es determinista, por lo que al entrenar varias instancias de ese modelo no se generan clasificadores exactamente iguales.
  - El *ensemble* se compone de modelos obtenidos con distintos métodos.
  - Cada modelo se entrena con un conjunto de datos diferente.
  - Cada modelo se entrena con un subconjunto de atributos diferente para el conjunto de datos.

Teniendo en cuenta estas distinciones a continuación se exponen los modelos de clasificación que se han probado en este proyecto.

Random Forest

El clasificador *Random Forest* es un *ensemble* que emplea aprendizaje supervisado mediante árboles de decisión, y se trata de uno de los más sencillos y utilizados. Para construir cada modelo simple emplea la técnica de *Bagging*, que consiste en entrenar cada modelo con un conjunto de datos del mismo tamaño que el original pero generado mediante muestreo uniforme con reemplazo (los datos pueden aparecer varias veces en el mismo conjunto). Mediante este mecanismo se reduce la varianza y se evita el sobreajuste.

Además, durante la construcción de los árboles que componen el *ensemble*, en cada nodo de decisión se tendrá en cuenta únicamente un subconjunto aleatorio de atributos del conjunto de datos.

Dado que todos los modelos del *ensemble* son independientes los unos de los otros, tienen la misma importancia, por lo que su voto tendrá el mismo peso a la hora de calcular la predicción final.

One-Class

La detección de anomalías One-Class consiste en entrenar el modelo con instancias de una sola clase de forma que a la hora evaluar nuevas instancias, devolverá información sobre si cada una pertenece o no a la clase basándose, de alguna forma, en su similitud con las instancias usadas en el entrenamiento.

Por ejemplo, si entrenamos el clasificador solo con instancias etiquetadas como «no crisis» y utilizamos datos de ambas clases para la predicción, esta puede devolver dos valores:

- La nueva instancia corresponde con la clase, por lo que se predice «no crisis».
- La nueva instancia no corresponde con la clase, por lo que se predice «crisis».

Durante el entrenamiento no se indica explícitamente a qué clase pertenece cada instancia, pero dado que todas las instancias de entrenamiento pertenecen a la misma, sí se le está proporcionando cierta información sobre la clase a la que pertenecen, y por lo tanto puede ser considerado como un algoritmo de aprendizaje supervisado.

AdaBoost.M1

Se trata de un *ensemble* para clasificación multi-clase que emplea aprendizaje supervisado. Emplea el método de *Boosting*, que a diferencia de *Bagging* es iterativo, y genera cada modelo del *ensemble* influenciado por el rendimiento del anterior. En este caso se busca que cada modelo dé más importancia a las instancias que son mal clasificadas por el modelo anterior. A diferencia de *Bagging* este mecanismo puede producir sobreajuste, sobre todo si el conjunto de datos tiene demasiado ruido.

Debido a la forma en la que se construyen los modelos simples que lo componen, a la hora de calcular la predicción final para una nueva instancia el voto de cada modelo tendrá un peso distinto en función de su precisión.

Rotation Forest

Rotation Forest [7] es también un *ensemble* para clasificación que emplea aprendizaje supervisado mediante árboles de decisión. No se basa en un

algoritmo iterativo como *Boosting*, por lo que los clasificadores simples pueden ser generados de forma paralela, y se trata de un método algo más complejo que los anteriores, basado en la aplicación del ya mencionado *PCA* a subconjuntos, preferiblemente disjuntos, de atributos de conjuntos de datos construidos mediante muestreo con reemplazo a partir del conjunto original.

3.6. Evaluación de patrones

En el contexto de este trabajo, la utilidad de los patrones encontrados corresponderá con la precisión del clasificador en sus predicciones. Dado que al clasificar instancias nuevas no conocemos a priori la clase a la que pertenecen, no podemos saber si la clasificación ha sido correcta, por lo que para calcular la precisión de un clasificiador se debe usar una partición de entrenamiento y test del conjunto original de datos disponibles:

- La **partición de entrenamiento** contiene la mayoría de los datos (por ejemplo, el 75 %), los cuales serán usados para entrenar el clasificador.
- El resto corresponden con la **partición de test**, que serán usados para comparar su clase real (su etiqueta) con la clase predicha por el clasificador entrenado.

La precisión corresponde con el porcentaje de acierto del clasificador para la partición de entrenamiento, pero en ocasiones, conviene usar métricas más complejas para medir el rendimiento de un clasificador. En este trabajo se ha trabajado con dos métricas distintas: el área bajo la curva ROC y la precisión media, pero antes de explicarlos es necesario presentar algunos conceptos básicos.

En un problema de clasificación binaria como este, si consideramos una instancia de «crisis» como un positivo (P) y una instancia de «no crisis» como un negativo (N) podemos definir los siguientes valores:

- Verdaderos positivos (VP): número de instancias de «crisis» que son clasificadas como «crisis».
- Falsos positivos (FP): número de instancias de «no crisis» que son clasificadas como «crisis».
- Verdaderos negativos (VN): número de instancias de «no crisis» que son clasificadas como «no crisis».

- Falsos negativos (FN): número de instancias de «crisis» que son clasificadas como «no crisis».

Estos valores se suelen visualizar en forma de matriz de confusión, la cual tiene la estructura de la figura 3.6.

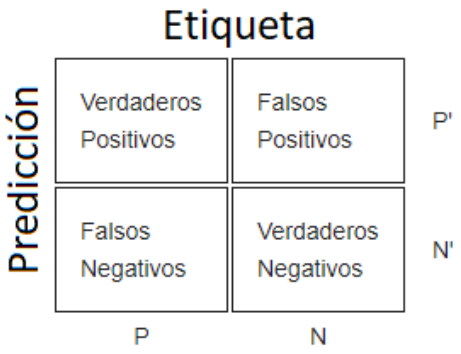

Figura 3.6: Disposición de la matriz de confusión.

Área bajo la curva ROC

A partir de los valores anteriores se pueden calcular los siguientes:

- **Sensibilidad o Razón de Verdaderos Positivos (VPR):**

$$VPR = \frac{VP}{P} = \frac{VP}{(VP+FN)}$$

- **Razón de Falsos positivos (FPR):**

$$VPR = \frac{VP}{P} = \frac{VP}{(VP+FN)}$$

Algunos clasificadores binarios, como los que se usan en este trabajo, además de predecir la etiqueta de la clase a la que creen que pertenece la instancia, son capaces de devolver la probabilidad entre 0 y 1 de que la instancia pertenezca a esa clase. Usando estas probabilidades podríamos clasificar las instancias en función de un umbral, por ejemplo, una instancia sería considerada «crisis» si la probabilidad de que pertenezca a esa clase es mayor o igual a 0.8 y «no crisis» si es menor.

El área bajo la curva ROC (*Receiver Operating Characteristic*) representa la relación entre la VPR y la FPR según se varía ese umbral. Se representará la VPR en el eje "y", y FPR en el eje "x". Cada punto corresponderá con los valores VPR y FPR resultantes de escoger cada uno de los posibles umbrales de clasificación, y la unión de esos puntos será lo que llamamos la curva ROC del clasificador.

Nos interesa que el VPR (sensibilidad) sea lo mayor posible y que el FPR (1-especificidad) sea lo menor posible, por lo que el clasificador óptimo corresponderá con aquel que presenta un área bajo la curva ROC (AUC) igual a 1, mientras que el peor clasificador será aquel con un AUC igual a 0.5, ya que ofrecería una predicción equivalente a lanzar una moneda al aire.

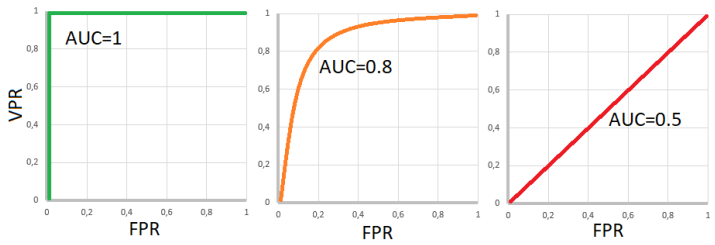

Figura 3.7: ROC. De izquierda a derecha: mejor caso, caso intermedio y peor caso.

Precisión media

Se definen las siguientes variables:

$$Precision = \frac{VP}{VP+FP}$$

$$Recall = \frac{VP}{VP+FN}$$

La precisión media (AV) se define como lo siguiente:

$$AV = \sum_{i=1}^{n}(Recall_i - Recall_{i-1})Precission_i$$

donde $Precission_i$ y $Recall_i$ son los valores de *Precision* y *Recall* calculados para el umbral de clasificación i-ésimo, y n el número de posibles umbrales. Cuanto mayor sea este valor, mejor se considera el clasificador.

3.7. Presentación del conocimiento

En este caso una posible presentación útil del conocimiento adquirido sería la representación de la probabilidad de pertenencia a la clase «crisis» de cada instancia ordenadas cronológicamente, es decir, cómo varía la probabilidad de crisis epiléptica del paciente con el tiempo en función del modelo. Esta será una de las gráficas que serán visibles en la aplicación de Android desarrollada.

Figura 3.8: Ejemplo de representación de la probabilidad de crisis en función del tiempo.

Técnicas y herramientas

En este apartado se presentan las técnicas metodológicas y las herramientas que se han usado en las distintas fases del desarrollo del proyecto.

4.1. Técnicas metodológicas

Scrum

Como se explica más detenidamente en el apartado de Planificación temporal del Apéndice A, para la planificación y desarrollo del proyecto se ha utilizado la metodología ágil *Scrum*, manteniendo el enfoque incremental de los *sprints* pero adaptándola al contexto de un trabajo con fines educativos.

KDD

En la fase de investigación se ha seguido el proceso de Descubrimiento de Conocimiento en Bases de Datos o *KDD*, dedicado a encontrar un modelo válido y útil en la medida en la que sirva para describir los patrones subyacentes de los datos. El término *KDD* suele ser empleado a menudo como sinónimo de minería de datos, pero esta corresponde en realidad con uno solo de los pasos del proceso. Se suele hablar de las siguientes fases englobados en el proceso de *KDD*:

- **Limpieza de datos:** Consiste en eliminar los datos inconsistentes o con ruido.
- **Integración de datos:** Se integran los datos de todas las fuentes disponibles en un formato uniforme y adecuado para los pasos posteriores.

- **Selección de datos:** Se trata de seleccionar solo aquellos datos relevantes para la tarea de análisis.
- **Transformación de datos:** Se realizan transformaciones y cálculos a partir de los datos en bruto con el fin de aplicar las técnicas de minería a las formas más apropiadas de los mismos.
- **Minería de datos:** Aplicación de técnicas para encontrar patrones subyacentes.
- **Evaluación de patrones:** Se estudia hasta qué punto los patrones encontrados son interesantes.
- **Presentación del conocimiento:** Consiste en ofrecer una representación comprensible y útil de los patrones y del conocimiento extraído.

Estos pasos no se aplican necesariamente de forma secuencial, ya que en muchos casos conviene volver a pasos anteriores tras la evaluación de los resultados del paso actual.

4.2. Herramientas en la fase de investigación

Anaconda

Anaconda es un administrador de paquetes y de entornos considerado un estándar para el desarrollo de minería de datos en lenguajes como Python y R. Al instalar Anaconda se tienen automáticamente disponibles más de 200 paquetes, además de ofrecer la posibilidad de añadir nuevos de forma sencilla.

Licencia: *New BSD License*

Jupyter Notebook

Los experimentos se han desarrollado en código Python distribuído en múltiples jupyter notebooks, ya que ofrecen un formato de estructuración y documentación del código muy adecuado para la investigación. *Jupyter Notebook* se encuentra disponible al instalar Anaconda.

Licencia: *Modified BSD License*

Scikit-Learn

Es la principal biblioteca empleada en la fase de investigación. Incluye modelos de clasificación, predicción y clustering de todo tipo y herramientas para entrenarlos, explotarlos y evaluarlos de forma sencilla. Está especialmente diseñada para operar con las bibliotecas *NumPy* y *SciPy*, y es compatible con *Pandas*.

Licencia: *New BSD License*

Weka

En algunas partes de la investigación se han usado modelos de Weka, otra plataforma para aprendizaje automático y minería de datos escrita en Java. Fue utilizada sobre todo para el entrenamiento de *ensembles* aplicados a conjuntos de datos desequilibrados.

Licencia: *GNU General Public License*

tsfresh

Es una biblioteca de Python dedicada al cálculo de grandes cantidades de características de series temporales. Permite calcular 64 tipos distintos de características y dispone de herramientas para filtrarlas en función de su relevancia. Es compatible con *Pandas*.

Licencia: *MIT License*

DEAP

Es el framework de python para computación evolutiva más utilizado. Se ha empleado como una de las alternativas para intentar encontrar la mejor selección de características de series temporales mediante un algoritmo genético, esperando que al aplicar este conjunto en la fase de minería se obtuviesen los mejores resultados posibles.

Licencia: *GNU Lesser General Public License v3.0*

tmux

Se trata un multiplexador de terminales que permite abrir varias sesiones simultáneamente y dejarlas corriendo en segundo plano. Esta herramienta ha resultado especialmente útil para la ejecución de los experimentos más

costosos en cuanto a tiempo de ejecución. Debido a que las ejecuciones de los experimentos se han llevado a cabo sobre un equipo de cómputo del grupo de investigación de los tutores mediante una conexión ssh que a menudo se cerraba en medio de un trabajo, ha sido necesario abrir una sesión de tmux en el equipo para cada una de estas ejecuciones de forma que, aunque se perdiera la conexión, el proceso siguiera corriendo en segundo plano y pudiéramos acceder a los resultados reactivando la sesión cuando el trabajo finalizase.

Licencia: *BSD License*

Otras bibliotecas relevantes

- *NumPy* y *Pandas* para la gestión, modificación y presentación de los datos.
- *Matplotlib* para la presentación gráfica de los resultados.

4.3. Herramientas en la fase de diseño de la aplicación

StarUML

Software de edición de diagramas UML utilizado en la fase de modelado.

Licencia: Propietaria aunque dispone de una demo gratuíta.

Material Design

Es una guía de estilo desarrollada por Google e integrada a partir de la versión *Lollipop* (5.0) de Android. Esta guía de estilos trata los elementos de la interfaz como elementos matariales, con unas dimensiones y una posición definida dentro del espacio (no solo en el plano, también en una tercera dimensión representada mediante el atributo *elevation*), propone una serie de dimensiones idóneas para cada tipo de elemento (textos, botones, tarjetas, etc.), y define la forma de generar el esquema de colores de la interfaz.

Pencil

Software de prototipado de interfaces gráficas. Permite incluir paquetes para incorporar elementos propios de *Material Design* a los prototipos, por

lo que proporciona una visión más próxima al aspecto final de las interfaces de la aplicación que otros tipos de prototipado.

Licencia: *GNU Public License version 2*

4.4. Herramientas en la fase de desarrollo de la aplicación

Android Studio

Es el entorno de desarrollo integrado oficial de Android, disponible para Windows, GNU/Linux y MacOS. Android Studio incluye, entre otras muchas cosas, un editor de código, un editor gráfico de *layouts*, emuladores para todas las versiones de Android existentes, y soporte para construcción automática con Gradle.

Licencia: *Apache License 2.0*

Gradle

Herramienta para la automatización de la construcción del software para proyectos Java. Es la herramienta soportada de forma oficial por Android.

Licencia: *Apache License 2.0*

Android Support Library

Es la biblioteca que gestiona la compatibilidad de funciones de versiones avanzadas con su equivalente en versiones anteriores de Android. Además, incluye *layouts*, elementos y utilidades que no están disponibles en el framework oficial. Aunque la biblioteca recomendada actualmente para este cometido es *AndroidX*, la cual incluye las mismas funcionalidades y algunas más, se ha escogido *Android Support Library* por su simplicidad y por su documentación clara y completa, lo que resulta de mucha utilidad cuando se desarrolla una aplicación Android por primera vez.

Licencia: *Apache License 2.0*

MPAndroidChart

Es una biblioteca para generación de gráficos en aplicaciones Android. En este caso, y aunque no está pensada para ello, se ha utilizado para

visualizar gráficas dinámicas cuyos datos se modifican en tiempo real. Para implementar esta funcionalidad se tuvo en cuenta también la biblioteca SciChart, al ser la biblioteca de referencia para generación de gráficos en tiempo real de Android, pero se descartó al tratarse de un software de pago.

Licencia: *Apache License 2.0*

Socket.IO-client Library

Es una biblioteca para gestión de comunicación mediante sockets para Java. El uso de esta biblioteca viene impuesto por la implementación de la API del servidor remoto, que gestiona el envío de datos en tiempo real de esta forma.

Licencia: *MIT License*

4.5. Otras herramientas generales

- **Git** como sistema de control de versiones distribuido.
- **GitHub** como plataforma para el *hosting* del repositorio del proyecto.
- **ZenHub** como extensión de GitHub para la gestión del proyecto basada en la metodología *Scrum*.
- **GitKraken** como cliente de Git mediante interfaz gráfica. (*GNU Public License*).
- **Overleaf** como editor colaborativo de LaTeX online para la generación del cuaderno de investigación conjunto.
- **TeXstudio** como editor de LaTeX para la generación de la memoria y los anexos. (*GNU General Public License v2*).
- **FileZilla** como aplicación para transferencia de ficheros. Soporta los protocolos FTP, SFTP y FTPS. (*GNU General Public License v2*).

Aspectos relevantes del desarrollo del proyecto

Este apartado pretende recoger los aspectos más interesantes del desarrollo del proyecto, comentados por los autores del mismo. Debe incluir desde la exposición del ciclo de vida utilizado, hasta los detalles de mayor relevancia de las fases de análisis, diseño e implementación. Se busca que no sea una mera operación de copiar y pegar diagramas y extractos del código fuente, sino que realmente se justifiquen los caminos de solución que se han tomado, especialmente aquellos que no sean triviales. Puede ser el lugar más adecuado para documentar los aspectos más interesantes del diseño y de la implementación, con un mayor hincapié en aspectos tales como el tipo de arquitectura elegido, los índices de las tablas de la base de datos, normalización y desnormalización, distribución en ficheros3, reglas de negocio dentro de las bases de datos (EDVHV GH GDWRV DFWLYDV), aspectos de desarrollo relacionados con el WWW... Este apartado, debe convertirse en el resumen de la experiencia práctica del proyecto, y por sí mismo justifica que la memoria se convierta en un documento útil, fuente de referencia para los autores, los tutores y futuros alumnos.

Trabajos relacionados

Este apartado sería parecido a un estado del arte de una tesis o tesina. En un trabajo final grado no parece obligada su presencia, aunque se puede dejar a juicio del tutor el incluir un pequeño resumen comentado de los trabajos y proyectos ya realizados en el campo del proyecto en curso.

Conclusiones y Líneas de trabajo futuras

Todo proyecto debe incluir las conclusiones que se derivan de su desarrollo. Éstas pueden ser de diferente índole, dependiendo de la tipología del proyecto, pero normalmente van a estar presentes un conjunto de conclusiones relacionadas con los resultados del proyecto y un conjunto de conclusiones técnicas. Además, resulta muy útil realizar un informe crítico indicando cómo se puede mejorar el proyecto, o cómo se puede continuar trabajando en la línea del proyecto realizado.

Bibliografía

[1] 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.

[2] Federación Española de Epilepsia. Qué es la epilepsia, 2013. [Internet; consultado 11-Junio-2019].

[3] 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.

[4] Usama M. Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI Magazine*, 17:37–54, 03 1996.

[5] 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.

[6] Sriram Ramgopal, Sigride Thome-Souza, Michele Jackson, Navah Ester Kadish, Iván Sánchez Fernández, Jacquelyn Klehm, William Bosl, Claus Reinsberger, Steven Schachter, and Tobias Loddenkemper. Seizure detection, seizure prediction, and closed-loop warning systems in epilepsy. *Epilepsy & behavior*, 37:291–307, 2014.

[7] 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.

[8] Scikit-Learn. Comparison of manifold learning methods, 2019. [Internet; descargado 14-junio-2019].

[9] Scikit-Learn. sklearn.decomposition.pca, 2019. [Internet; descargado 14-junio-2019].

[10] Alexandros T Tzallas, Markos G Tsipouras, Dimitrios G Tsalikakis, Evaggelos C Karvounis, Loukas Astrakas, Spiros Konitsiotis, and Margaret Tzaphlidou. Automated epileptic seizure detection methods: a review study. In *Epilepsy-histological, electroencephalographic and psychological aspects*. IntechOpen, 2012.

[11] Wikipedia. Filtro de savitzky–golay — wikipedia, la enciclopedia libre, 2015. [Internet; descargado 14-junio-2019].

[12] Wikipedia. Filtro de butterworth — wikipedia, la enciclopedia libre, 2016. [Internet; descargado 14-junio-2019].

[13] Wikipedia. Serie temporal — wikipedia, la enciclopedia libre, 2018. [Internet; descargado 14-junio-2019].