



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 – 26 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.

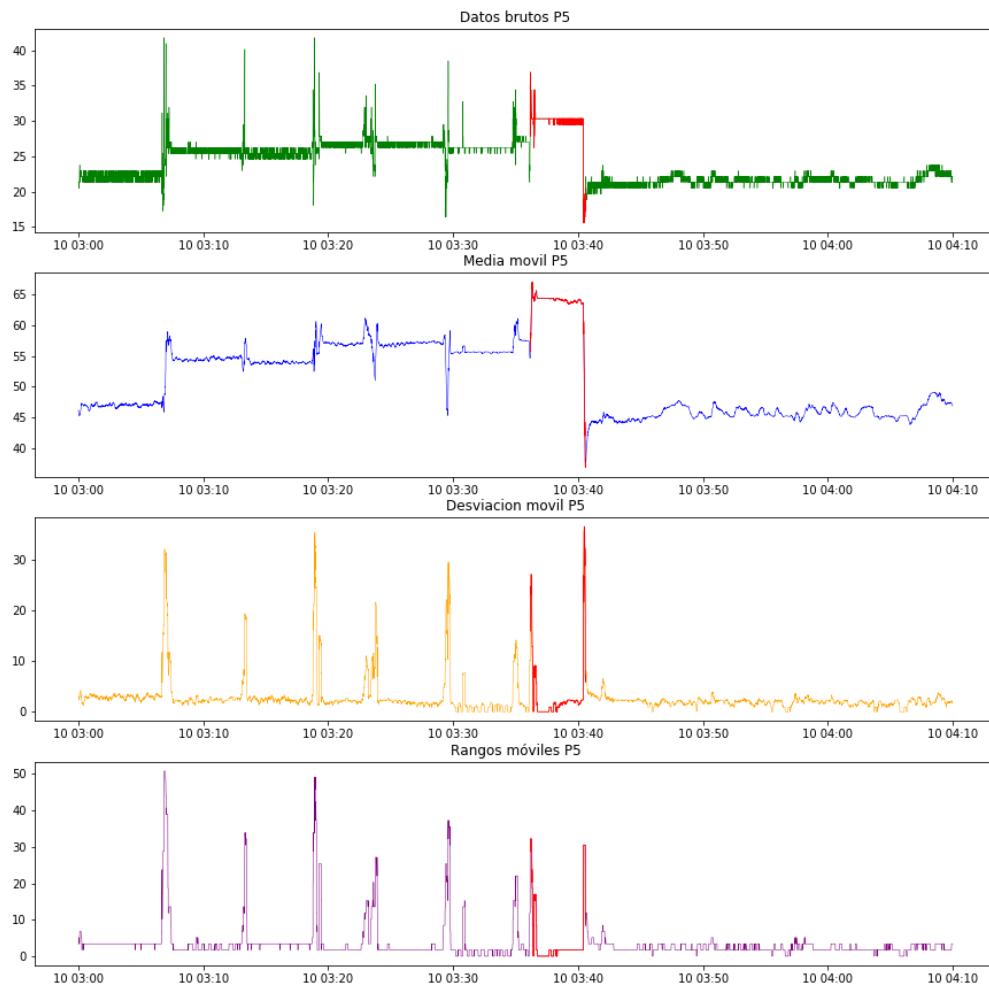


Figura 1.1: Señales de los datos brutos y estadísticos asociados al tubo de presión P5. Estimación de la crisis epiléptica en rojo.

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

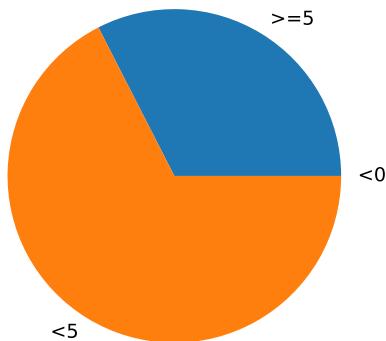
El filtrado es otra técnica empleada para eliminar el ruido, hemos empleado varias filtros como el *butterworth* [16] con valores $N=3$ y $Wn=0.05$ cuyo resultado se puede ver en la Figura 2.2a. Independientemente del tipo de filtro se aplican con la función `scipy.signal.filtfilt`. También se han probado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

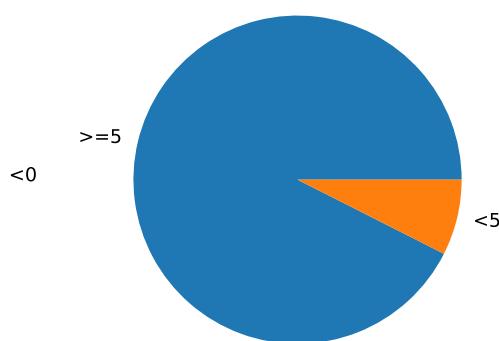
Para hacer más pequeño el conjunto de datos eliminamos aquellos tubos cuya variabilidad sea menor a un umbral, por defecto ese umbral de de 0.5.

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

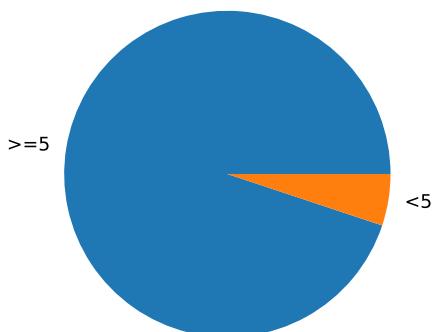
²Este ha sido el criterio para determinar al valor 5 como el mínimo para un valor real de presión



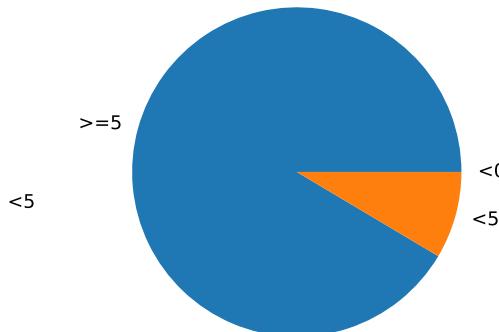
(a) Proporción de ruido en tubo de presión 1



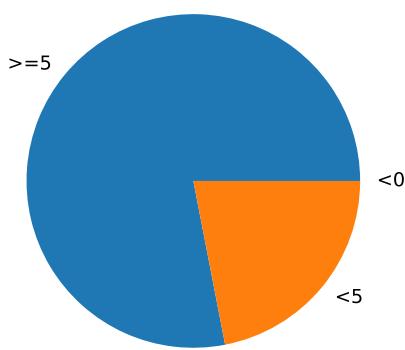
(b) Proporción de ruido en tubo de presión 2



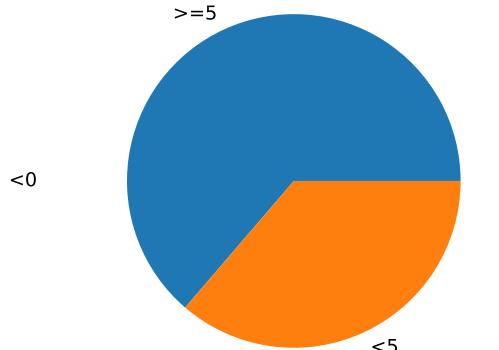
(c) Proporción de ruido en tubo de presión 3



(d) Proporción de ruido en tubo de presión 4

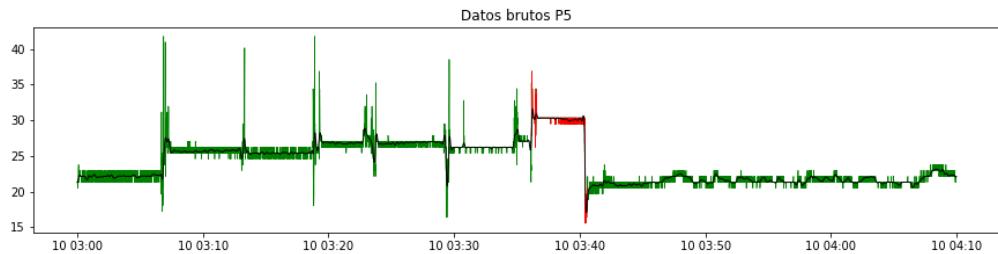


(e) Proporción de ruido en tubo de presión 5

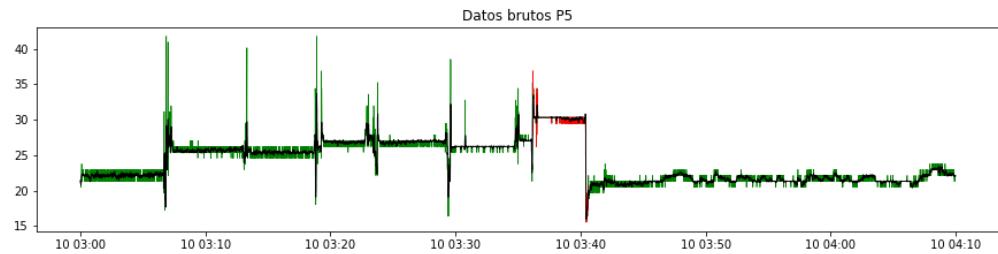


(f) Proporción de ruido en tubo de presión 6

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)

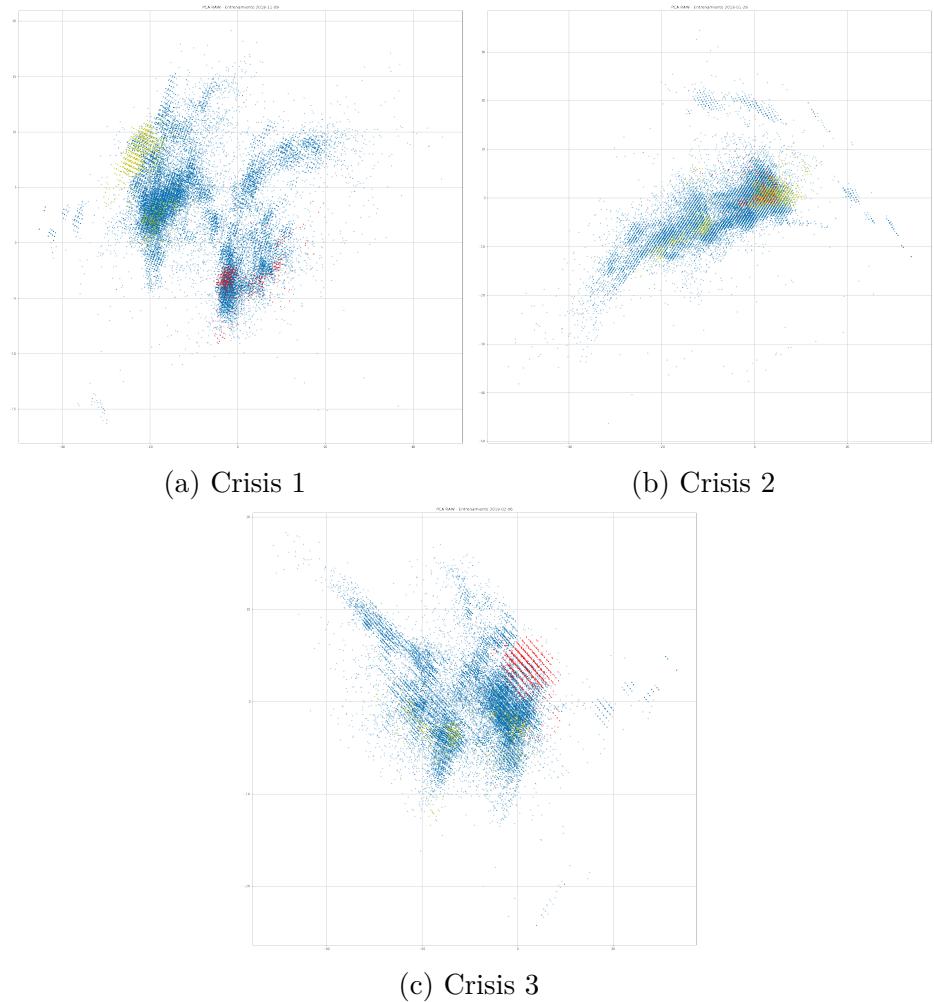


Figura 4.1: PCA ajustada a cada crisis

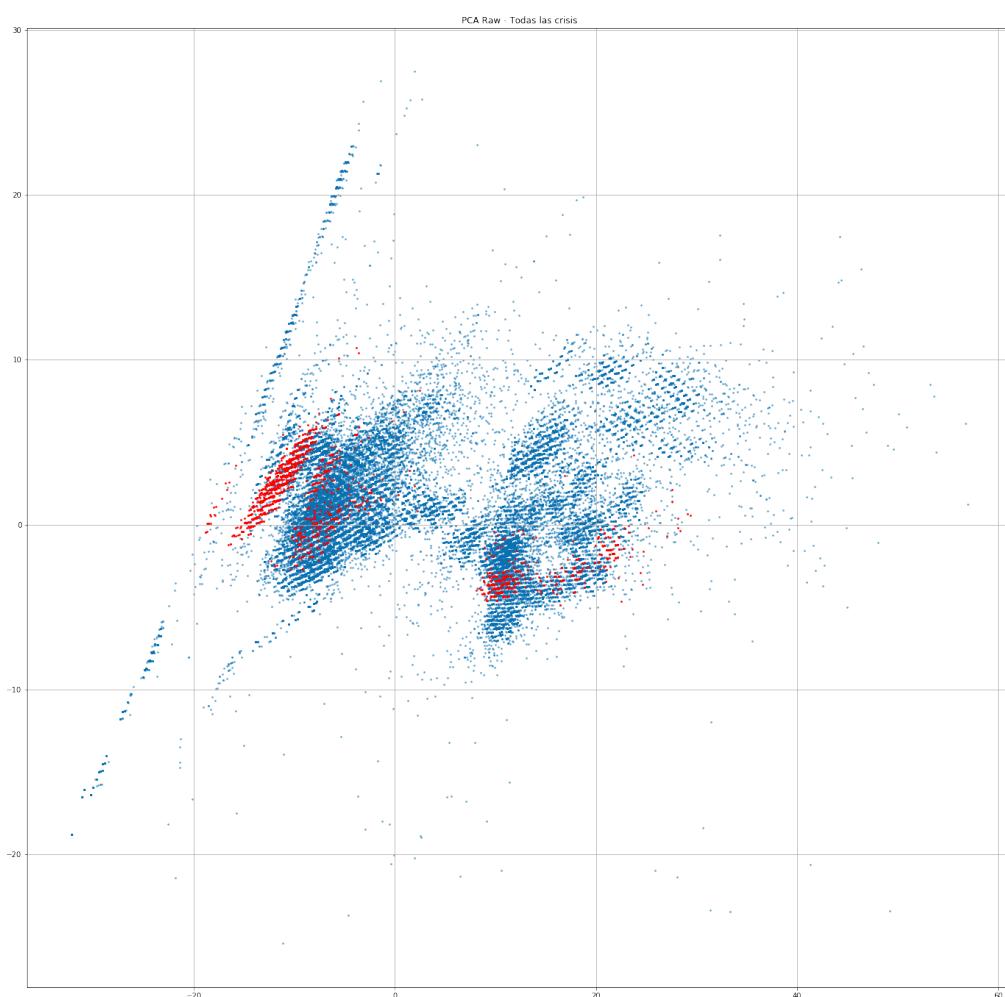


Figura 4.2: PCA ajustada con todas las crisis

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.

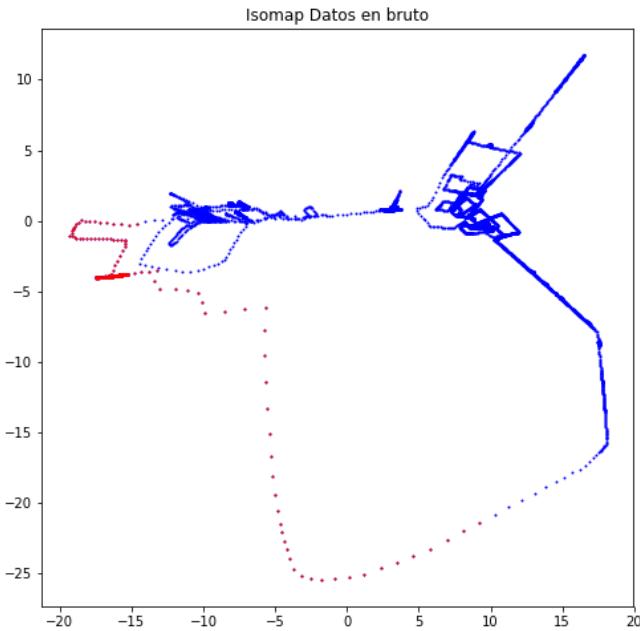


Figura 5.1: Proyección Isomap 2D de los datos brutos de presiones. Crisis epiléptica en rojo.

- 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).

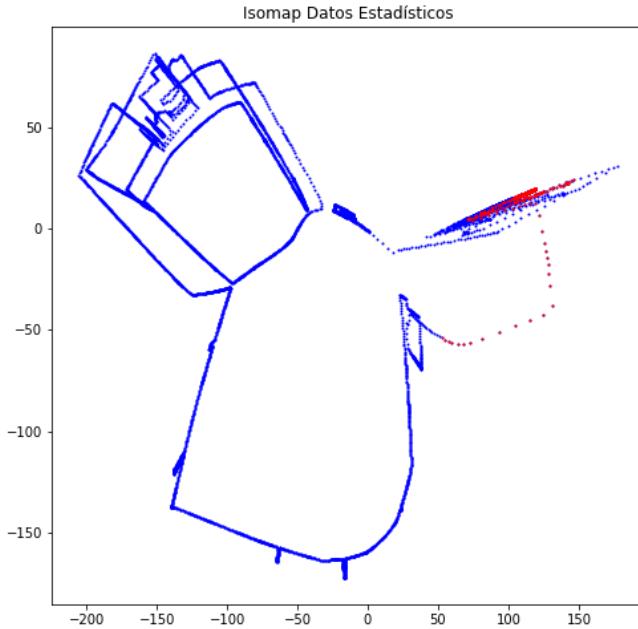


Figura 5.2: Proyección Isomap 2D de los datos estadísticos de presiones. Crisis epiléptica en rojo.

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`.

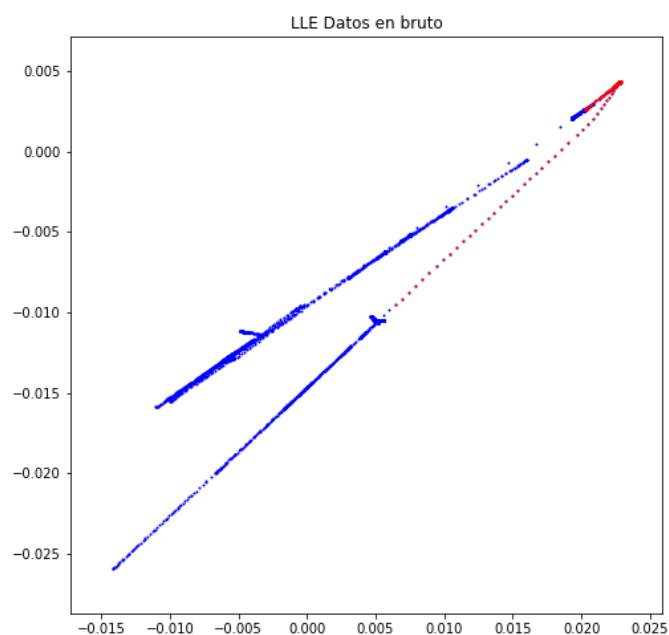


Figura 5.3: Proyección LLE 2D de los datos brutos de presiones. Crisis epiléptica en rojo.

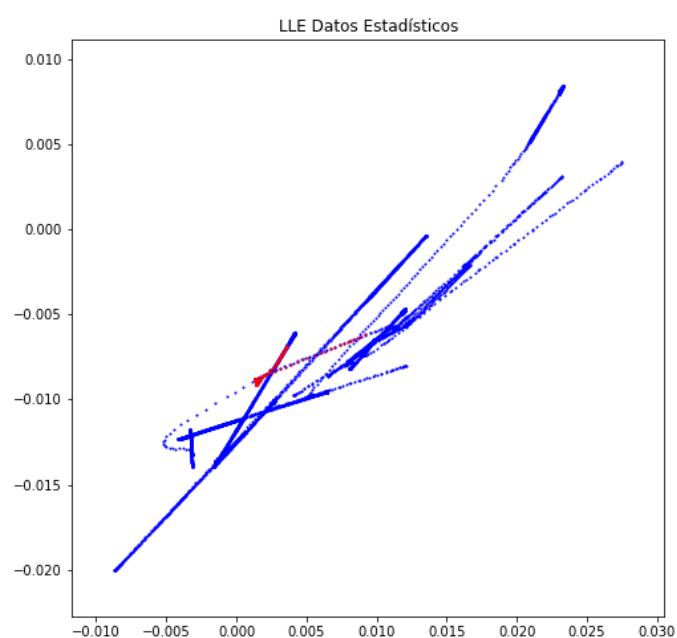


Figura 5.4: Proyección LLE 2D de los datos estadísticos de presiones. Crisis epiléptica en rojo.

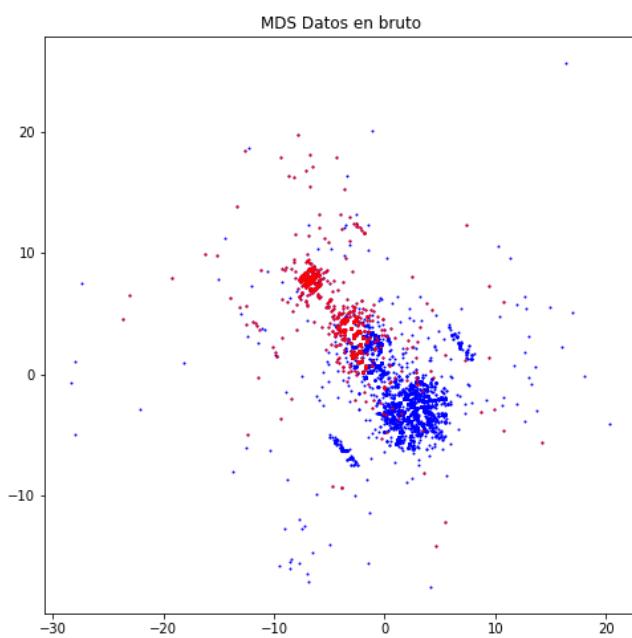


Figura 5.5: Proyección MDS 2D de los datos brutos de presiones. Crisis epiléptica en rojo.

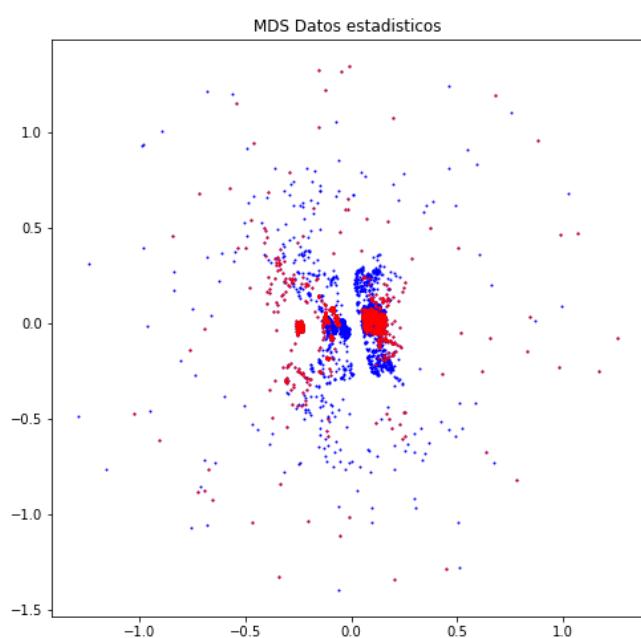


Figura 5.6: Proyección MDS 2D de los datos estadísticos de presiones. Crisis epiléptica en rojo.

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 (auc1¹).
3. Se entrenará el clasificador con los datos del día de la segunda crisis y testeado con los de la primera.
4. Se calculará el rendimiento con los resultados del testeo (auc2²).
5. Se calculará el rendimiento final como la media entre auc1 y auc2. Buscamos que el rendimiento final sea lo mayor posible.

8.4. Primera exploración, métrica=ROC

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

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

¹AUC del testeo entrenando con la primera crisis

²AUC del testeo entrenando con la segunda crisis

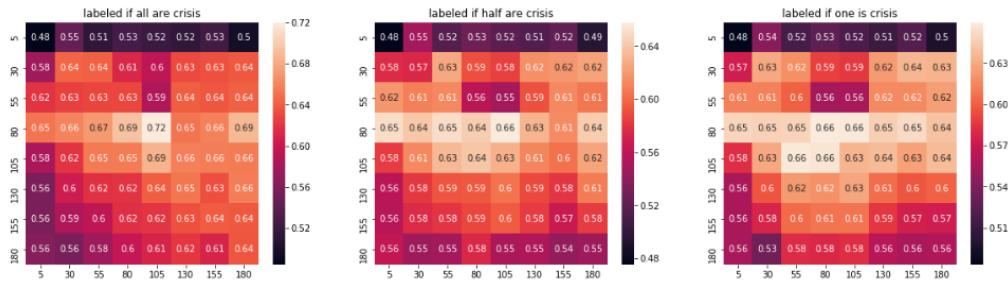


Figura 8.1: Mapa de calor de las áreas bajo la curva calculadas en la primera exploración con la métrica=ROC.

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

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

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

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

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

8.5. Segunda exploración, métrica=ROC

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

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

CAPÍTULO 8. RANDOM FOREST – MEDIA Y DESVIACIÓN MÓVILES

36

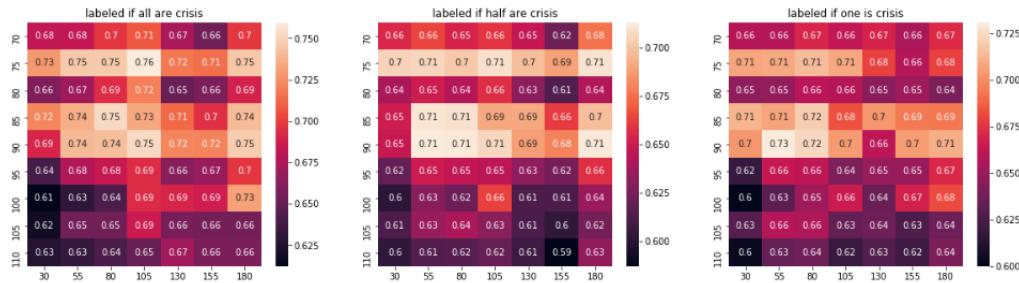


Figura 8.2: Mapa de calor de las áreas bajo la curva calculadas en la segunda exploración con la métrica=ROC.

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

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

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

8.6. Tercera exploración, métrica=ROC

Finalmente, centramos la exploración un poco más:

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

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

- Ventana para la media = 90

8.7. PRIMERA EXPLORACIÓN, MÉTRICA=PRECISION-RECALL 37

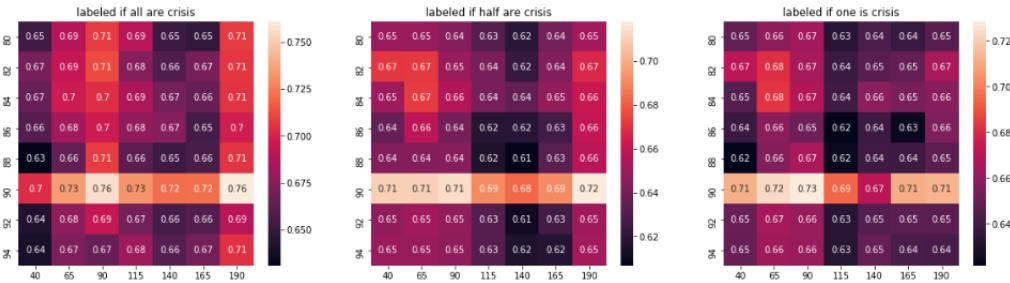


Figura 8.3: Mapa de calor de las áreas bajo la curva calculadas en la tercera exploración con la métrica=ROC.

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

8.7. Primera exploración, métrica=Precision-Recall

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

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

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

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

CAPÍTULO 8. RANDOM FOREST – MEDIA Y DESVIACIÓN

38

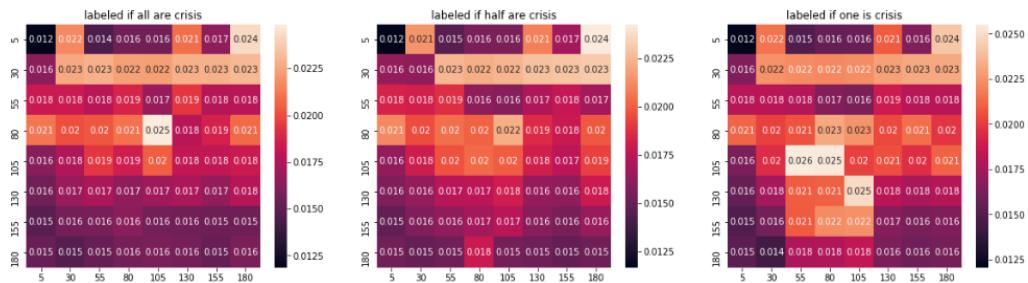


Figura 8.4: Mapa de calor de las áreas bajo la curva calculadas en la primera exploración con la métrica=Precision-Recall.

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.

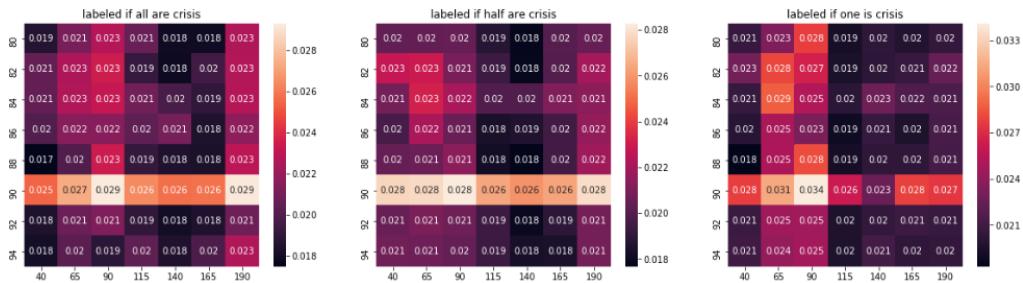


Figura 8.6: Mapa de calor de las áreas bajo la curva calculadas en la tercera exploración con la métrica=Precision-Recall.

- 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`.

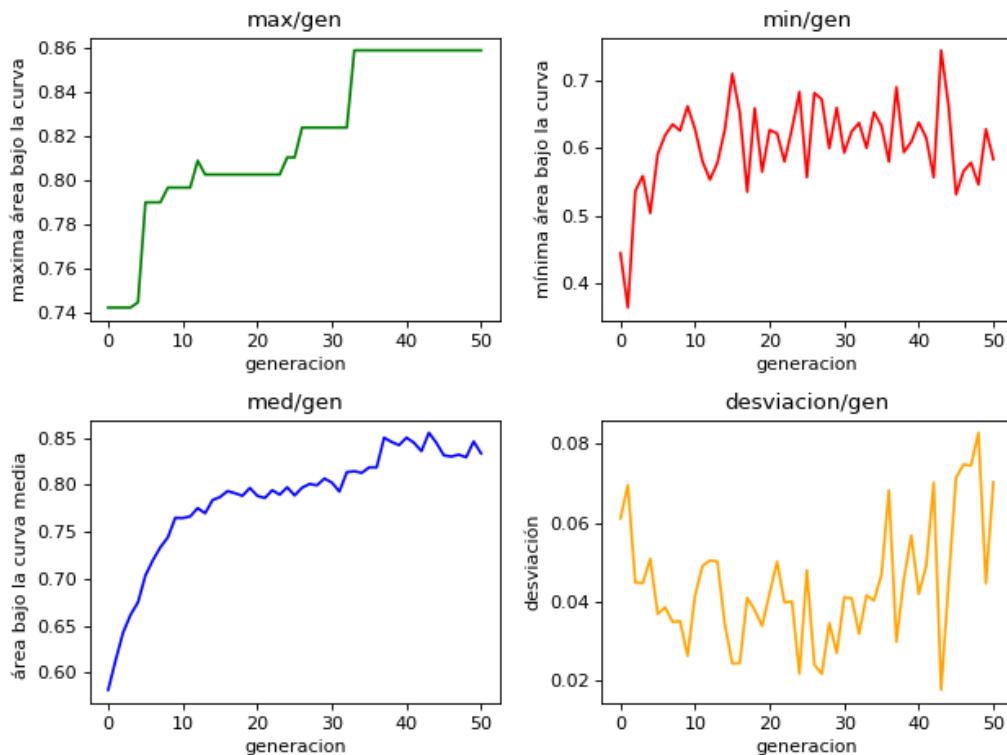


Figura 9.1: Gráficas de la evolución de las poblaciones en el algoritmo genético con la métrica=ROC.

- `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

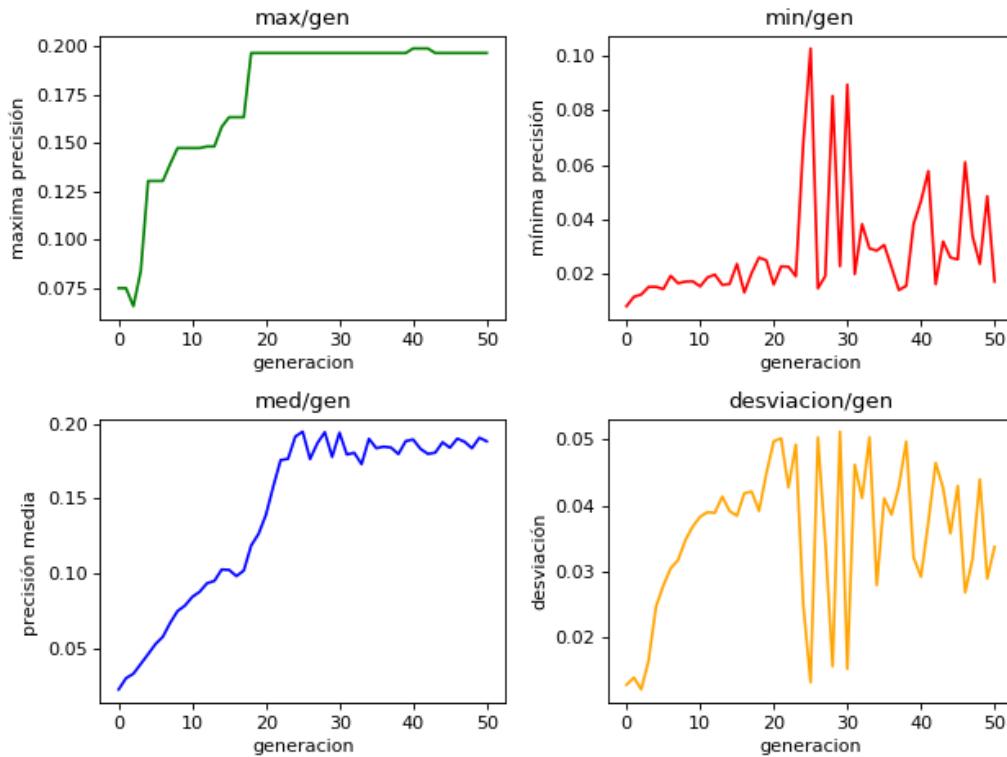


Figura 9.2: Gráficas de la evolución de las poblaciones en el algoritmo genético con la métrica=Precision-Recall.

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

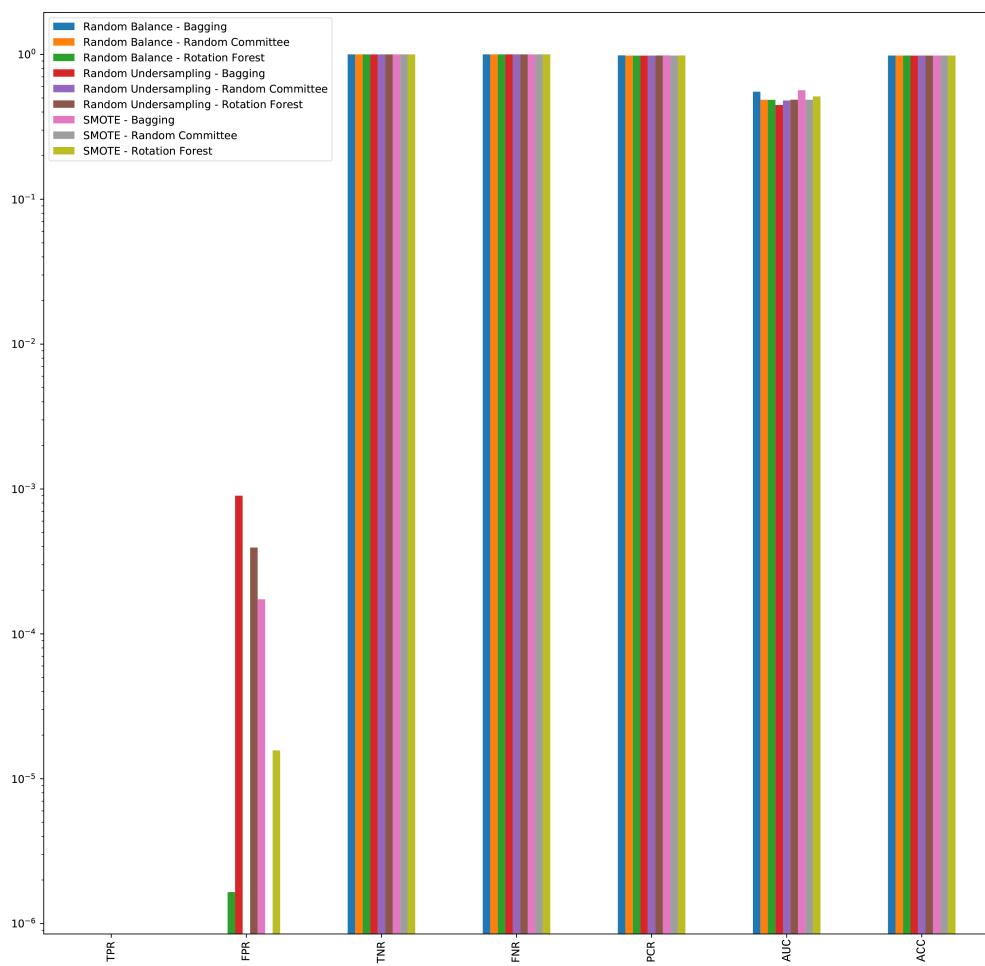


Figura 10.1: Métodos para desbalanceados - Entrenamiento con la primera crisis, testeo con la segunda crisis.

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.

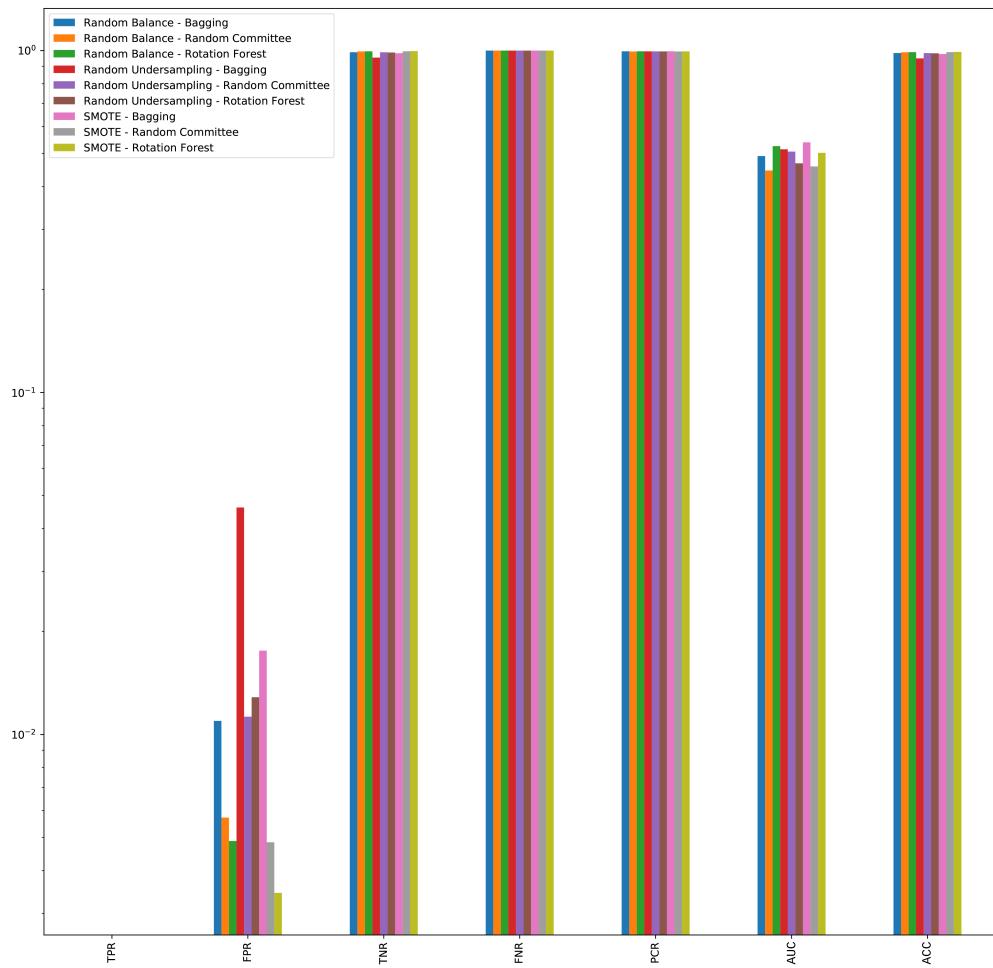


Figura 10.2: Métodos para desbalanceados - Entrenamiento con la segunda crisis, testeo con la primera crisis.

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.