Machine Learning para la clasificación de actividades físicas

Juan Francisco Teran, Mateo Olaya

Mayo 2024

1 Metodología

El problema consiste en desarrollar un clasificador que sea capaz de predecir la actividad física que una persona realiza, basándose en señales recopiladas por acelerómetros incorporados en dispositivos móviles como teléfonos celulares y relojes inteligentes. Este desafío implica procesar y analizar datos complejos generados por diferentes personas al hacer diversas actividades, que incluyen caminar, correr, sentarse, pararse, subir escaleras y bajar escaleras. El objetivo es diseñar un modelo de Machine Learning eficiente que, al interpretar las variaciones y patrones únicos en algunas de las propiedades más relevantes extraídas de las series temporales de los movimientos, pueda identificar con buena exactitud su clase correspondiente.

Para resolver este problema del mundo real, probamos utilizando algoritmos de Machine Learning como AdaBoost y Artificial Neural Network para la clasificación, K-Means para Clustering, y PCA y UMAP para visualización de datos, implementando modelos de ML de Ensemble Learning, Deep Learning y Unsupervised Learning. Además, decidimos no usar otras características adicionales, a las variables que ya venían en el Dataset de entrenamiento "metadata.csv", para solucionar el problema.

2 Análisis Exploratorio de Datos

Manipulamos el conjunto de datos original por medio de distintas técnicas para llegar al Dataset final que se utilizó para los modelos de Machine Learning.

2.1 Correlación entre variables

En este caso particular al tener un total de treinta y dos características (32) acordamos que un paso importante antes de entrar de lleno al modelado, era definir cuales de aquellas características eran las más importantes, o las que más impacto iban a tener en el modelo final, para de este modo no solo potenciar el impacto de las que finalmente quedaron en el modelo, sino para mitigar o

en este caso eliminar el que iban a tener las que decidimos remover del modelo final.

Para ello lo que hicimos fue entrenar un modelo Random Forest con todas las características del dataset, una vez entrenado, se obtiene la importancia que cada característica tuvo en el modelo final que se puede visualizar a modo de gráfico.

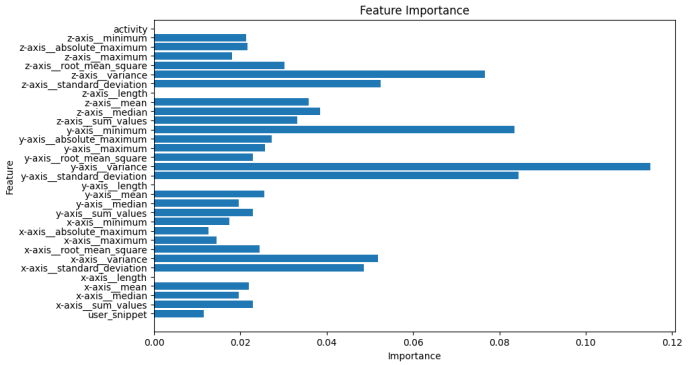

Figure 1: Importancia de las características del dataset en el modelo

Finalmente llegamos a la conclusión de que el paso a seguir era eliminar las siguientes características dado su bajo impacto en el modelo: user snippet, x-axis length, y-axis length, z-axis length, x-axis maximum, y-axis maximum, z-axis maximum.

2.2 Rangos de valores de las diferentes variables

Detectamos que el Dataset presenta valores sospechosos que no tienen coherencia con el contexto particular. Esto lo evidenciamos gracias a las variables cuantitativas "axis length" para cada eje del acelerómetro, las cuales nos permiten notar que algunas señales no están descritas en 100 puntos definidos como deberían.

Variable	**Valor mínimo**	**Valor máximo**
x-axis sum values	-1083.17	1098.5600000000002
x-axis median	-14.92	14.73
x-axis mean	-10.8317	10.985600000000002
x-axis length	3.0	100.0
x-axis standard deviation	0.015657586	12.903504173673136
x-axis variance	0.00024516	166.50041996000004
x-axis root mean square	0.064451532	15.040870054621172
x-axis maximum	-7.59	19.91
x-axis absolute maximum	0.11	19.91
x-axis minimum	-19.61	8.2

y-axis sum values	-1077.18	1159.45
y-axis median	-14.995	14.385000000000002
y-axis mean	-10.7718	11.5945
y-axis length	3.0	100.0
y-axis standard deviation	0.018335485	11.704012506401384
y-axis variance	0.00033619	136.98390874999998
y-axis root mean square	0.1647088340071655	14.36356637468564
y-axis maximum	-3.68	20.04
y-axis absolute maximum	0.34	20.04
y-axis minimum	-19.61	9.96
z-axis sum values	-795.48	952.65
z-axis median	-8.065	9.53
z-axis mean	-7.9548	9.5265
z-axis length	3.0	100.0
z-axis standard deviation	0.0196967	10.729626589495089
z-axis variance	0.00038796	115.12488674999996
z-axis root mean square	0.1669222	10.948550086655311
z-axis maximum	-7.59	19.61
z-axis absolute maximum	0.34	19.8l
z-axis minimum	-19.8	9.47

Table 1: Rangos de las diferentes variables para "metadata"

2.3 Valores perdidos

Aparentemente no hay valores faltantes en el Dataset. No obstante, gracias al rango de las variables "axis length" para cada eje (X, Y y Z), podemos observar que algunas señales del acelerómetro no están descritas en 100 puntos definidos. Esto es problemático ya que una serie representada en una longitud de cantidad menor no debería tener información suficiente para poder identificar su actividad correspondiente, por ende, decidimos considerarlos como datos perdidos.

Cantidad faltante (#)	Porcentaje faltante (%)
66	0.03268621236133122

Table 2: Valores perdidos para "metadata"

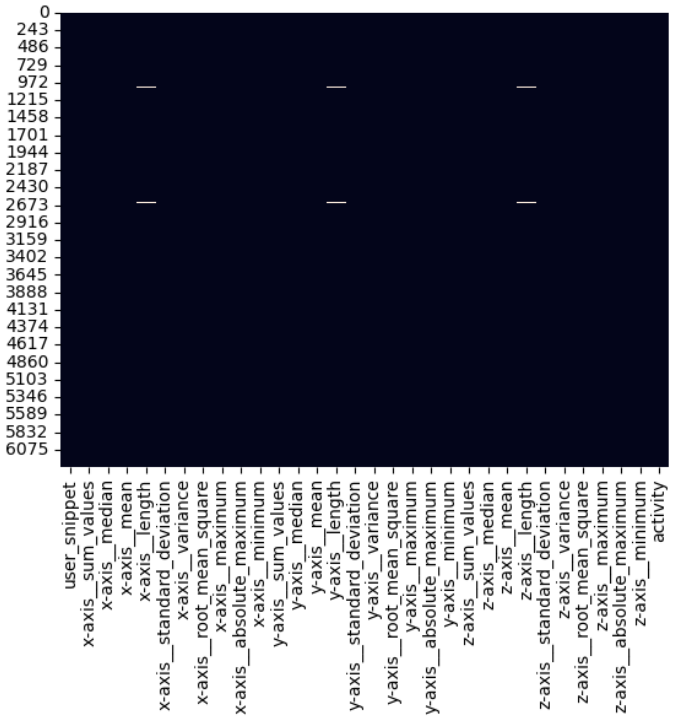

Figure 2: Mapa de calor de valores perdidos para "metadata"

Debido a que los datos faltantes no se pueden reconstruir ya que la imputación podría no tener sentido o ser irreal, veíamos apropiado eliminar las correspondientes observaciones.

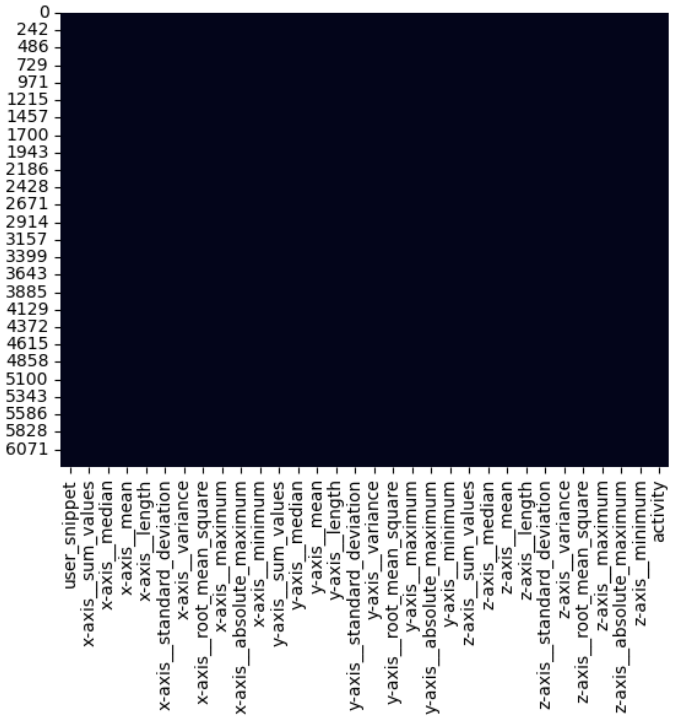

Figure 3: Mapa de calor de valores perdidos tras eliminación para "metadata"

No obstante, finalmente decidimos no hacer este preprocesamiento de los datos ya que el archivo "metadata kaggle" igual contenía algunos segmentos con una longitud menor a 100 que era obligatorio predecir. Por tal motivo, dejamos que el modelo pueda aprender también de aquellos valores que consideramos perdidos para poder predecir otros desconocidos que también lo son y así no tener problemas para hacer las respectivas predicciones de Kaggle.

2.4 Normalización de datos

Para ello, utilizamos la función "StandardScaler", la cual nos asegura que los datos estén centrados alrededor de cero y tengan una desviación estándar de uno.

Variable	**Valor mínimo**	**Valor máximo**
x-axis sum values	-2.679937557	2.361989355927624
x-axis median	-3.692548404	3.2861706418549734
x-axis mean	-2.676624067	2.359414063240567
x-axis length	-28.24277029	0.052423333227264494
x-axis standard deviation	-1.642351476	3.255847536422925

x-axis variance	-0.92364728	5.052979995337914
x-axis root mean square	-1.903641312	2.933740891
x-axis maximum	-2.722744203	1.447271324
x-axis absolute maximum	-2.355334477	1.249977544926663
x-axis minimum	-1.736112874	2.5774858426630862
y-axis sum values	-4.488622919	1.0810387104521268
y-axis median	-5.291421243	1.695949779270012
y-axis mean	-4.494172918	1.078128214868841
y-axis length	-28.24277029	0.052423333227264494
y-axis standard deviation	-1.903460775	2.5652307357276642
y-axis variance	-1.13002467	3.738192407293633
y-axis root mean square	-3.910411121	1.892438343025236
y-axis maximum	-4.283859242	0.6985598637766638
y-axis absolute maximum	-4.141106575	0.6905930956890195
y-axis minimum	-2.235499148	1.8069904671301293
z-axis sum values	-3.804568731	4.2597044083922935
z-axis median	-3.523766713	4.2038197420412144
z-axis mean	-3.800628762	4.254165835095978
z-axis length	-28.24277029	0.052423333227264494
z-axis standard deviation	-1.907885087	3.4857065556565625
z-axis variance	-1.052150041	5.515040665882265
z-axis root mean square	-2.311426675	3.5063701824614504
z-axis maximum	-3.986435815	1.854982968960965
z-axis absolute maximum	-2.696989438	1.7982712271877062
z-axis minimum	-2.092994178	3.0367601674432105

Table 3: Rangos de las diferentes variables tras normalización para "metadata"

Dado que desarrollamos modelos de ML con ANN, se requiere y exige escalar las variables antes de aplicar el algoritmo. Además, estandarizar los datos fue importante al realizar una selección secuencial de características en el Dataset que mejoró el rendimiento de los modelos neuronales. Por último, normalizar las características fue relevante para la visualización de datos.

2.5 Visualización de datos

Decidimos usar 2 algoritmos de visualización de datos: Principal Component Analysis (PCA) y Uniform Manifold Approximation and Projection (UMAP).

La distribución de los datos proyectados por PCA nos permitió observar principalmente 4 grupos, en los cuales en 3 de ellos se pueden diferenciar sus clases (círculo amarillo: Sitting (5), círculo café: Standing (4), círculo rojo: Jogging (0)). Sin embargo, en el círculo rosado vimos que es una zona del mapa muy mezclada entre datos que corresponden a Walking (1), Downstairs (2) y Upstairs (3), donde es importante destacar principalmente la dificultad de

una separación clara entre Downstairs y Upstairs, lo cual puede deberse a las similitudes en los movimientos y señales.

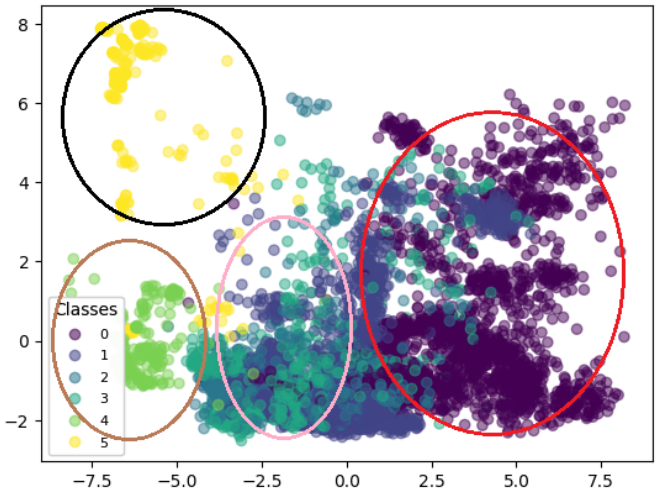

Figure 4: PCA: Visualización de datos para "metadata""

Por otro lado, UMAP no parece tener una separación muy buena de las clases. No obstante, no es mucho problema ya que PCA ofreció bastante información.

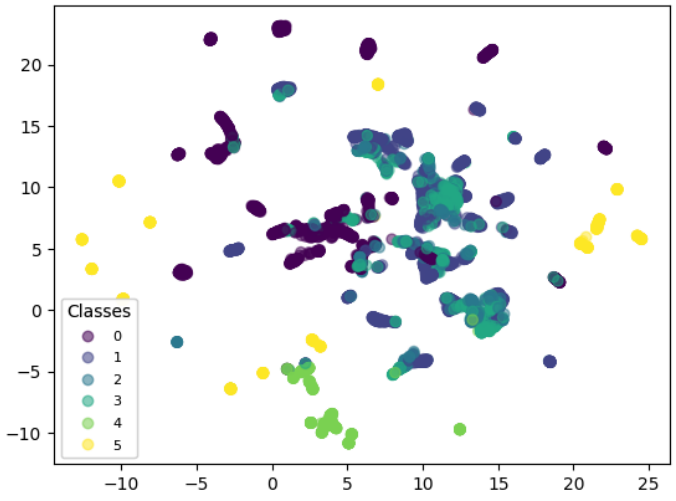

Figure 5: UMAP: Visualización de datos para "metadata"

3 Modelado

En nuestra estrategia de diseño de modelos, la idea fue buscar maximizar el desempeño de los modelos. Para ello, utilizamos 3 conjuntos de datos distintos y 10-Fold Cross-Validation.

Datasets necesarios:

- TrainingData: Conjunto de datos para entrenar el modelo. Se compone del 90% de "metadata".
- ValidationData: Conjunto de datos para la optimización. Se compone del 10% de "metadata".
- TestData: Conjunto de datos para evaluar la generalización. Corresponde a "metadata test".

10-Fold Cross-Validation:

1. Separar el Dataset en ValidationData (10%) y TrainingData (90%).
    - El número de iteraciones define el porcentaje de los casos que se usarán para ValidationData sobre la totalidad del conjunto de datos.
2. Iterar el paso 1 sin repetir casos que ya fueron tomados anteriormente para ValidationData.

Decidimos realizar 10-Fold CV porque este permite que los modelos no queden sesgados gracias a un tamaño adecuado del TrainingData.

3.1 AdaBoost

3.1.1 Selección de modelos

El siguiente gráfico de conteo permite visualizar que las clases están desbalanceadas, por lo que decidimos utilizar Balanced Accuracy como métrica de desempeño para medir y puntuar el modelo más óptimo que encontramos.

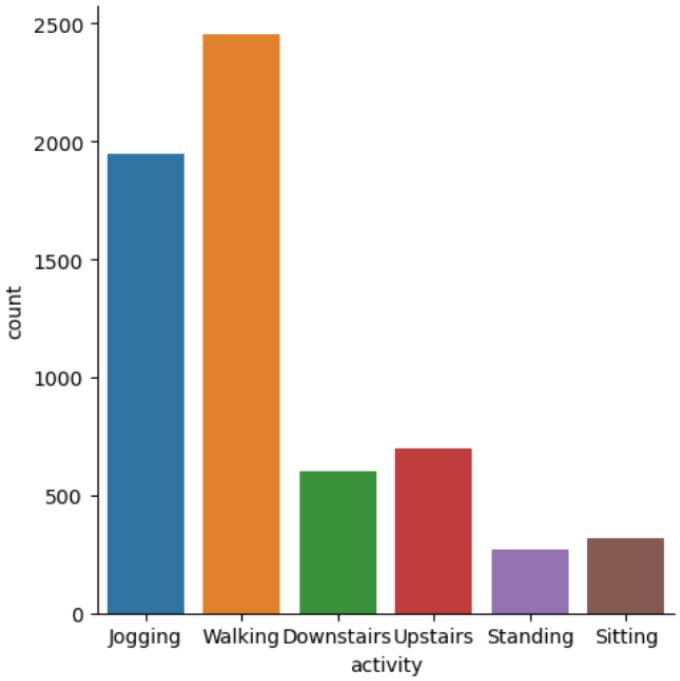

Figure 6: Distribución de la salida "activity"

Para el problema de optimización del modelo, decidimos buscar valores de los hiperparámetros por medio del algoritmo GridSearch ya que queríamos garantizar que se probaran todas las combinaciones que se nos ocurrieran. Para ello:

1. Utilizamos TrainingData, ValidationData y 10-Fold Cross-Validation para seleccionar nuestro mejor modelo, de todos los posibles, definiendo sus hiperparámetros ideales.
   (a) Definimos los valores que queríamos probar para cada hiperparámetro que consideramos relevante.
   (b) Creamos un modelo probando una combinación basada en los valores definidos en el paso (a) para los hiperparámetros que se tienen.
   (c) Estimamos el desempeño obtenido al modelo del paso (b) aplicando 10-Fold Cross-Validation con TrainingData y ValidationData.
   (d) Iteramos los pasos (b) y (c) hasta probar todas las combinaciones de los diferentes valores de hiperparámetros definidos en (a) y haciendo 10-Fold Cross-Validation de cada modelo.
   (e) Sobre todas las combinaciones, seleccionamos aquella que crea el mejor modelo, es decir, el que tiene mayor media de Balanced Accuracy para sus 10 iteraciones distintas de Cross-Validation.

- Tras una búsqueda muy amplia de los mejores hiperparámetros, lo mejor que pudimos encontrar aproxima Train Accuracy al 80%, de tal manera que consideramos que el modelo no hace Underfitting.

2. Utilizamos el TrainingData para reentrenar el modelo seleccionado en 1.(e).

Hiperparámetros	Mejores hiperparámetros	Train Accuracy
estimator: [weak_learner], learning_rate: range(100, 150, 10), n_estimator:[1.0, 0.5, 0.1, 0.01], algorithm: ['SAMME', 'SAMME.R']	estimator: weak_learner, learning_rate: 0.1, n_estimator: 140, algorithm: 'SAMME'	79.7%

Table 4: Ajuste de hiperparámetros para "mdl_ada"

Hiperparámetros	Mejores hiperparámetros	Train Accuracy
min_samples_leaf: range(3,18,3), min_samples_split: range(2,18), max_depth:[1, 3, 7, 9, 12, 15, 18, 20], max_features: [3, 4, 5, 6, None], ccp_alpha: [0.0, 0.001, 0.005,0.010, 0.015, 0.020, 0.025, 0.030, 0.035, 0.05], min_impurity_decrease: [0.0, 0.1]	min_samples_leaf: 20, min_samples_split: 17, max_depth: 20, max_features: 5, ccp_alpha: 0.0, min_impurity_decrease: 0.0	75.0%

Table 5: Ajuste de hiperparámetros para "weak_learner"

3.1.2 Evaluación de modelo

Para tener la capacidad de saber el desempeño del modelo seleccionado cuando tiene nuevos casos que predecir correctamente, podemos usar el TestData.

1. Evaluar y probar el modelo seleccionado reentrenado para la parte de generalización.

   - Si había una desviación del 5% entre la Train Accuracy y la Test Accuracy, entonces regulábamos los hiperparámetros con el fin de reducir la complejidad del modelo y controlar el Overfitting.

- El desempeño de nuestro modelo subóptimo es con una Test Accuracy del 82.1% y una Kaggle Accuracy equivalente al 80.0%. La matriz de confusión para nuestro TestData se muestra a continuación:

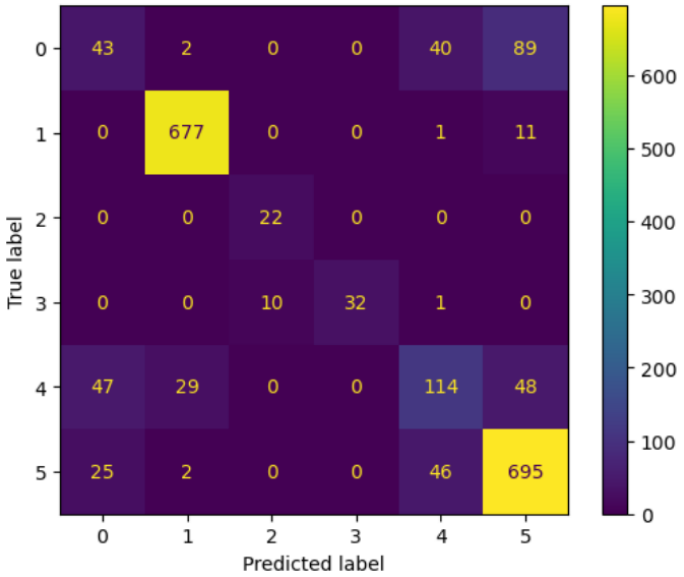

Figure 7: Matriz de Confusión de "mdl_ada" para TestData

Nombre	Test Accuracy	Kaggle Accuracy
mdl_ada	82.1%	80.0%

Table 6: Accuracy para "mld_ada"

3.2 ANN

3.2.1 Selección del mejor modelo

En el caso del modelo implementado con Aritificial Neural Network lo primero que se realizó fue encontrar los mejores hiperparámetros para crear el modelo, esto fue posible gracias a un Randomized Search, inicialmente definimos una gama de valores que podían tomar los hiperparámetros, un total de 10 iteraciones y como se menciono antes, un total de 10 folds para la validación cruzada. Los resultados de esta búsqueda se detallan a continuación.

hiperparámetro	**Valor**
activation	relu
alpha	0.03047724259507192

batch_size	64
hidden_layer_sizes	(50, 100)
learning_rate	constant
learning_rate_init	0.002326496115986653
momentum	0.9422017556848528
solver	adam

Table 7: Mejores hiperparámetros para ANN obtenida a través de Random-Search

Una vez obtenidos los mejores hiperparámetros, se crea el modelo de ANN con dichos parámetros y un numero de iteraciones igual a cuatrocientos (400), sin embargo, durante el entrenamiento y testeo pudimos apreciar que el 'accuracy' era muy inferior al esperado, pese a que estos hiperparámetros fueron los que mejor resultado estaban brindando entre las varias iteraciones realizadas con el Randomized Search, era evidente que lo que fallaba era la definición de la gama de parámetros que asignamos para la búsqueda, es por ello que decidimos cambiar los valores de 'hidden_layer_sizes' teniendo en cuenta que buscábamos un modelo con capas 'simétricas' y que debía clasificar entre seis (6) clases diferentes optamos mediante experimentación por usar un modelo con capas (233, 70, 30, 6, 30, 70, 233).

Para la selección del modelo tuvimos en cuenta que en cada entrenamiento los pesos en las capas de la red neuronal cambiaban pese a tener los mismos hiperparámetros, motivo por el cual se obtenían métricas distintas en cada iteración, del mismo modo era importante tener presente el hecho de que el comportamiento de un modelo con el set de test iba a ser diferente al que tendría con el set de kaggle, es por ello que iterando constantemente se seleccionaron un total de cinco (5) modelos de ANN que fueron enviados a la plataforma obteniendo así al final un 'score' de **0.82493** con nuestro modelo de Artificial Neural Network.

Para medir los modelos de Artificial Neural Networks como primer opción usamos la curva de perdida o ´loss curve´en base a las iteraciones o 'epochs' que tuvo el modelo durante su entrenamiento.

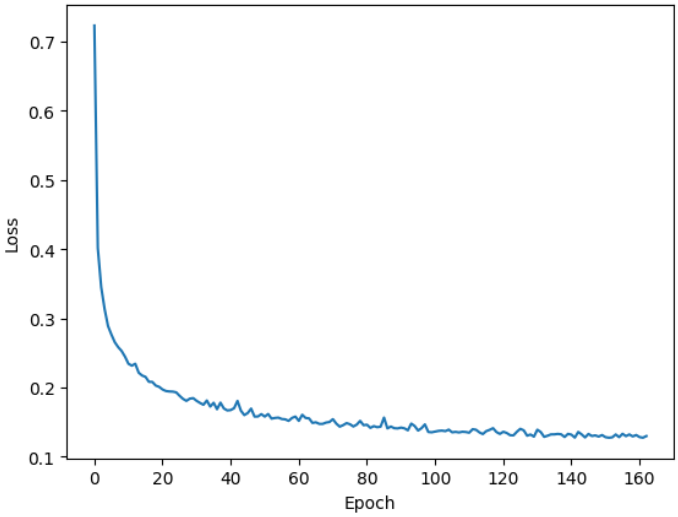

Figure 8: 'Loss Curve' del modelo de ANN

El comportamiento de la curva es el esperado, sin embargo esperábamos y nos hubiera gustado una curva más consistente y estable, a simple vista se puede notar bastante ruido lo que puede indicar un comportamiento errático del modelo entre cada iteración.

3.2.2 Evaluación de los modelos

El primer criterio que usamos para evaluar los modelos fue el 'accuracy' con respecto a 'metadata_test' en este punto el resultado consideramos fue de un nivel óptimo al obtener consistentemente modelos de un 'accuracy' superior a 0.8.

Por último, tambien usamos como criterio la matriz de confusión para evaluar los modelos de Artificial Neural Networks.

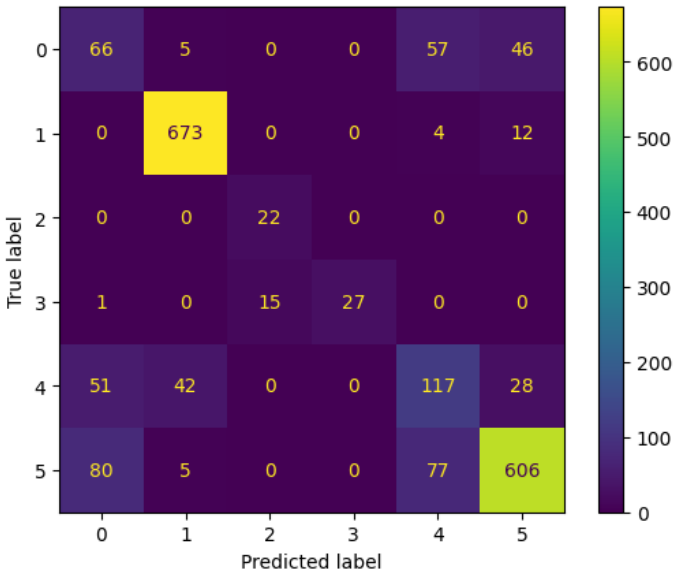

Figure 9: Matriz de confusión del modelo de ANN

4 Discusión

- De los algoritmos, el mejor fue el de ANN porque con AdaBoost, el modelo no se entrenaba muy bien y no tenía muy buenos resultados, pese a hacer una búsqueda en cuadrícula demasiado exhaustiva y demorada. Esto debe ser porque, aunque es posible que hayamos encontrado buenos valores para los hiperparámetros, el dataset se hubiese podido pulir más haciendo reducción de dimensionalidad con PCA, teniendo en cuenta los buenos resultados que arrojó la visualización del mismo algoritmo (y reduciendo el ruido que se ve en la misma) o con autoencoders, aprovechando su no-linealidad contraria a PCA.

- Consideramos que este trabajo podría mejorarse si, en lugar de haber asumido que el dataset estaba preparado para el análisis solo con hacer normalización de los datos, hubiéramos aprovechado el Exploratory Data Analysis para entender el dataset y darnos cuenta de que los modelos implementados con aprendizaje supervisado podrían funcionar mejor haciendo más técnicas de preprocesamiento de datos, sobretodo aquellas que involucran algoritmos de aprendizaje no supervisado. Además, es posible que extrayendo otras características de "signals.csv", como por ejemplo el tiempo que demora cada segmento en alcanzar su máximo o mínimo, el dataset original tuviera datos de mejor calidad para el modelado.

- El modelo que desarrollamos lo pudimos obtener gracias a la ayuda de la búsqueda aleatoria para los valores de los hiperparámetros de la red neuronal. Esto es sumamente relevante, ya que aunque Grid Search garantiza que se prueben todas las combinaciones que se quieran, consume mucho tiempo, sobretodo en algoritmos como ANN, en los cuales hay muchos distintos hiperparámetros importantes. Random Search es más rápido en comparación y permite explorar todo el espacio de búsqueda sin límites.

- A pesar de que probamos demasiados modelos diferentes, consideramos que no alcanzamos el desempeño más alto, tal vez porque nos faltó aplicar algoritmos de reducción de la dimensionalidad y evaluar la posibilidad de enriquecer convenientemente el dataset original con nuevas variables extraídas de las series temporales del archivo "signals".

- La selección de características de entrenamiento para el modelo de Machine Learning es fundamental, pese a que a través del uso de RandomForest visualizamos el impacto o importancia de las características en el modelo, consideramos que intentar con un modelo de regresión logistica y un selector de características secuencial pudo haber dado un resultado que aportará a la consecución de un mejor desempeño en la competencia.

5 Apéndice

1. Código que evidencia el trabajo realizado como archivo Jupyter Notebook.