Exploración de modelos de aprendizaje automático para la clasificación *online* de mediciones de sensores MOX-Humedad-Temperatura

Carlos Anívarro
carlos.anivarro@estudiante.uam.es

Daniel Barahona
daniel.barahonam@estudiante.uam.es

Daniel Cerrato
daniel.cerrato@estudiante.uam.es

David T. Garitagoitia
david.garitagoitia@estudiante.uam.es

F.A.A - Escuela Politécnica Superior – Universidad Autónoma de Madrid

Abstract. Se proponen varios métodos para la clasificación de estímulos en base a medidas de sensores MOX. Se busca predecir con la mayor fiabilidad posible y en un plazo corto de tiempo si los sensores se hallan expuestos a un estímulo (vino o banana), o se encuentran ante un ambiente neutro. Un estudio comparativo entre Redes Neuronales y Random Forest reveló elevadas métricas de rendimiento de clasificación para este último, con un *accuracy* en test del 99%. Se provee además el código utilizado para el desarrollo de esta exploración.

Técnicas del curso empleadas: *missing values*, Random Forest, Matriz de Confusión.

1 Introducción

La tarea a resolver es la siguiente: una serie de sensores MOX (*metal-oxide*) han tomado medidas del ambiente cada pocos segundos en una habitación. De manera aleatoria, en ciertos momentos se ha expuesto dichos sensores a estímulos de vino o banana, y en otros se ha mantenido un ambiente neutro.

El objetivo del problema es, dada una serie de medidas tomadas por los sensores, distinguir de manera precisa y fiable cuándo se está ante un ambiente neutro, y cuándo ante un estímulo (y en cuyo caso, determinar si es "vino" o "banana"). Además, se quiere poder aplicar la solución a entornos *online*, esto es, con datos en tiempo real. Por tanto, la respuesta ha de poder darse en base a no muchas medidas, y en un plazo de tiempo adecuado.

El resto de este paper se organiza como sigue: la Sección 2 expondrá un análisis exploratorio de los datos de partida. Los atributos y el dataset construido para los experimentos se presentarán en la Sección 3. La Sección 4 expondrá los modelos de aprendizaje automático evaluados con el dataset. Por último, los resultados de estas evaluaciones se discutirán en la Sección 5.

2 Análisis exploratorio de los datos

El conjunto de datos de partida [1] consiste en 100 "series" de medidas de cada sensor para cada instante de tiempo. Se dispone de diez sensores: 8 MOX (`R1-R8`) más uno de humedad

(`Humidity`) y otro de temperatura (`Temp.`). Estas "series" pueden pertenecer a una de 3 clases: *vino, banana* o *background*.

Las series disponen de medidas antes, durante y después de la colocación del estímulo, excepto cuando la etiqueta es *background* (entonces, toda la serie corresponde al ambiente neutro). Un fichero de metadatos indica cuánto dura la exposición de los sensores al estímulo ("ventana" de exposición). De alguna manera, la ventana indica también cuándo la "etiqueta cambia" de *background* a la del estímulo y viceversa.

La Figura 1 muestra la representación gráfica de dos series, para ilustrar las características comentadas (el intervalo de exposición al estímulo se marca entre líneas azules):

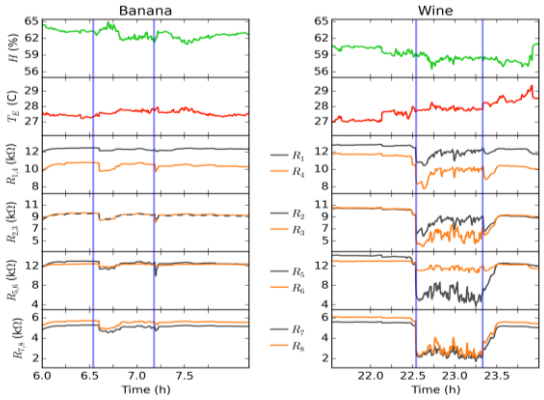

Figura 1: Representación gráfica de dos series con "banana" *(nº17, izq.) y* "vino" *(nº 19, der.)*

Un estudio preliminar desveló la siguiente distribución en tercios de las clases en las series originales (36 *vinos*, 33 *bananas* y 31 *background*):

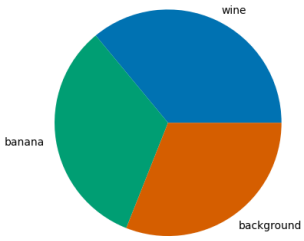

Figura 2: Distribución de clases en los datos de partida

Además, las ventanas temporales de exposición al estímulo (atributo *dt* de los metadatos) tienen una media de 0.7516 horas, y una desviación estándar de 0.428. Es decir, la mayoría de las series exponen los sensores al vino o la banana durante menos de una hora. Esto, junto con la preferencia de predecir con datos en tiempo real, nos indica que deberemos trabajar con ventanas del orden de pocos minutos, para poder cubrir los periodos de exposición en suficiente detalle.

El objetivo de esta primera parte del proyecto es desarrollar un *dataset tradicional* en base a los datos originales. Esto quiere decir que, antes de enviar los datos a un algoritmo de aprendizaje automático, deberemos primero estructurarlos en un formato de atributos (características) y clase.

Visto el análisis de los datos de partida, hemos notado lo siguiente:

- **Desbalanceo de clases**: se dice que un conjunto de datos presenta desbalanceo cuando el tamaño de una clase es significativamente desproporcionado al de otras clases. Esto tiene el peligro de inducir un sesgo en los modelos, que aprenderán mejor sobre las instancias pertenecientes a la clase mayoritaria. En el conjunto de partida, aparte de las series que son exclusivamente *background*, las que presentan un estímulo aun así disponen de mucho tiempo de ambiente neutro antes y después de la colocación del mismo, haciendo que las medidas de *background efectivas* superen el 80%.

- ***Missing Values*:** la serie 95 no aparece en el conjunto de datos originales. Como veremos más adelante, el dataset definitivo no distingue entre los identificadores originales (atributo *id*), por lo que no es necesario mitigar esta ausencia.

- En algunas series **las ventanas de exposición** que marca el fichero de metadatos **no están centradas** completamente con respecto a lo que marcan los sensores respecto a la aparición del estímulo. Por ejemplo, la Figura 3 muestra cómo el inicio de la ventana se adelanta ligeramente a lo que marcan los sensores para la serie 17, de tipo *banana*:

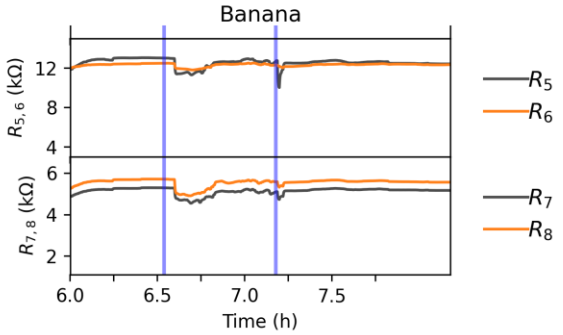

Figura 3: Ejemplo de ventana de exposición "descentrada"

Es por ello que se han tomado las siguientes decisiones a la hora de construir el dataset usado en los experimentos:

- **Ignorar las medidas fuera de las ventanas de exposición**: cada entrada de los metadatos contiene un atributo *dt* que indica cuánto dura la ventana de exposición al estímulo (incluso cuando este nunca aparece). Por tanto, se tomarán en cuenta los datos entre tiempo 0 y *dt*. De esta manera, tendremos una cantidad de datos de *background* más balanceada respecto a la cantidad de *bananas* y *vinos*.

- No obstante, una prueba preliminar demostró que los clasificadores son incapaces de reconocer las trazas de *background* en las series pertenecientes a *vino* o *banana*, posiblemente debido a que la presencia del estímulo queda patente en ciertas variables ambientales aún después de su retirada. Es decir, el dataset necesita reflejar tanto los *background* "puros" como los que han sido expuestos a un estímulo.

  Por tanto, se tomarán en cuenta los datos en la ventana de exposición (entre tiempo 0 y *dt*), **añadiendo un margen de 10 minutos a ambos lados de dicha ventana**. Así, pese a que tendremos una cantidad de datos de *background* algo desbalanceada respecto a la cantidad de *bananas* y *vinos*, esto beneficiará a la larga a reconocer las trazas de background en una serie expuesta al estímulo. (Un beneficio adicional de este margen es que soluciona el fenómeno de las "ventanas de exposición descentradas", pues logra abarcar todos los "saltos" en las gráficas, aunque estos se encuentren fuera de 0-*dt*).

- **Ignorar las medidas de humedad y temperatura**: Se tenía la hipótesis de que los estímulos propuestos no generan variaciones notables en estas magnitudes, al menos durante el tiempo de exposición. Es intuitivo pensar que la poca evaporación que se dé en una copa de vino no va a alterar significativamente la humedad ambiente, ni su temperatura. Un estudio estadístico[1] sobre el conjunto de datos ya construido confirmó la hipótesis. Por ello, y para tratar de optimizar los tiempos de predicción, se ha decidido desestimar estos dos atributos.

3 Atributos propuestos

El conjunto de datos final parte de las decisiones anteriores, con las siguientes consideraciones:

- El objetivo de lograr un reconocimiento del estímulo en tiempo real (*online*) sólo puede lograrse teniendo en cuenta ventanas de tiempo cortas para los datos que lleguen a los sensores. Como los intervalos de exposición al estímulo rondan la hora, se ha decidido establecer una ventana temporal de 5 minutos.

- La presencia de los estímulos *vino* y *banana* produce una alteración significativa en las lecturas de los sensores MOX. Por ello, para cada medida del dataset de partida, consideraremos medidas estadísticas que reflejen esa alteración en los 5 minutos anteriores. En concreto, consideramos la media y desviación estándar de los últimos 5 minutos para cada dato.

Con esto, se transformó el conjunto de datos de partida en un dataset que, aparte de mantener los 8 atributos originales (`R1-R8`, sin `Temp.` ni `Humidity`), añade la media (`_mean`) y la desviación típica (`_dev`) de todos ellos para los datos de la misma serie (mismo *id*) en la ventana de los 5 minutos anteriores al tiempo del dato. Es así como pasamos de un dataset de 10 atributos a uno de 24, ya que para cada sensor MOX original estamos preservándolo, y añadiendo las dos medidas

[1] Se muestran las gráficas de dicho estudio estadístico en el Anexo A, pero no es necesario verlas para la comprensión del informe.

estadísticas de la ventana previa (e ignorando la Humedad y la Temperatura). El nuevo conjunto de datos comprende 367.259 muestras.

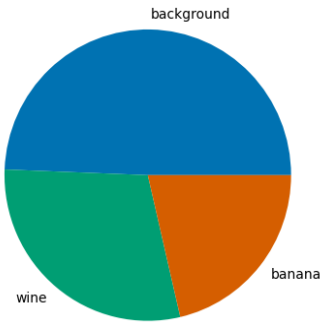

Figura 4: Distribución de clases en el dataset construido

La *Figura 4* muestra la distribución de clases en el dataset construido. Como podemos ver, la clase *background* muestra cierta preponderancia frente a la de *vino* y *banana*. Pese a ello, no consideraremos que el dataset presenta sesgo hacia esta clase, ya que como se ha mencionado, dentro de esas medidas catalogadas como *background*, están tanto las series puramente *background* como los márgenes de trazas de las series con estímulo. Es decir, ese *background* puede considerarse que aporta variedad al aprendizaje.

4 Modelos de clasificación

Con el conjunto de datos definitivo ya construido, se valoraron distintas opciones de clasificadores, para ver cuál cumplía con los estándares de precisión y rendimiento más apropiados a los requisitos del problema. Como objetivo secundario, cabe mencionar que el estudio original [1] obtenía un *accuracy* en test del 81%. Sería deseable intentar superar dichas métricas. Se ha hecho uso de dos clasificadores pertenecientes a la librería *Scikit-learn* [2]:

- *Random Forest*: es una apuesta interesante por varios motivos: el aprendizaje trata de reducir el fallo de los últimos árboles, como si estos aprendiesen de sus predecesores para encontrar una mejor división del espacio de atributos. En segundo lugar, la clasificación por método de "votación" reduce bastante la posibilidad de error puesto que se agregan las decisiones de un "jurado". Durante la fase de "tuneado de hiperparámetros" se ha realizado una criba eliminando las configuraciones que superan los 0.6 segundos de clasificación por muestra. Así la mejor combinación de valores es: construir datasets con el 100% del número de muestras (mediante *Bootstrap*), usar máximo 8 atributos para dividir el espacio, y un total de 21 estimadores (árboles).

- *Red Neuronal*: Un aspecto interesante de estos clasificadores es que las interacciones entre neuronas pueden generar combinaciones de atributos que reflejen mejor las dependencias y fortalezas entre ellos (durante su entrenamiento, estas neuronas se irán haciendo "más importantes"). Es interesante también el hecho de que su coste de clasificación es bastante bajo. El "tuneado de hiperparámetros" en este caso concluye en

que la mejor arquitectura de red era 4 capas ocultas con (24, 20, 16, 12) neuronas respectivamente. Como función de activación se emplea la ReLU (*Rectified Linear Unit*). La tasa de aprendizaje inicial se fija a 0.0001 y se ajusta en modo "adaptativo" en función de la *loss function*. Como optimizador se emplea el SGD (*Schotastic Gradient Descent*).

Para el análisis de los modelos de clasificación se ha utilizado validación simple, en la que se divide el dataset en una porción de *train* (70% de las muestras, es decir unas 257.000) y *test* (el 30% restante, unas 110.000). Los datos de *test* no se usaron para entrenar los modelos.

Se tomaron como métricas de rendimiento el *Accuracy*, la Precisión y el *F1-score*. El motivo de tomar esta última es debido a que el *accuracy* puede no mostrar toda la realidad de los datos cuando existe desbalanceo de clases entre ellos (como es nuestro caso). Por su parte, la *F1-score* ofrece una visión global del rendimiento del modelo sin verse afectada por estos desajustes en número de muestras. Como última medida, también se tomaron tiempos de clasificación para el objetivo de llevar el modelo a clasificación en tiempo real (*online*).

5 Discusión de resultados

La Tabla 1 muestra las métricas de rendimiento de clasificación para los dos modelos empleados, usando las configuraciones de hiperparámetros óptimas descritas en la Sección 4.

MODELO	ACCURACY	PRECISIÓN	F1-SCORE	TIEMPO DE CLASIFICACIÓN (segs)
Rand. Forest	0.999355	0.999355	0.999355	0.594
NN	0.92	0.91	0.91	0.652

Tabla 1: Métricas de rendimiento para los modelos

Adicionalmente se aportan las matrices de confusión asociadas a cada modelo:

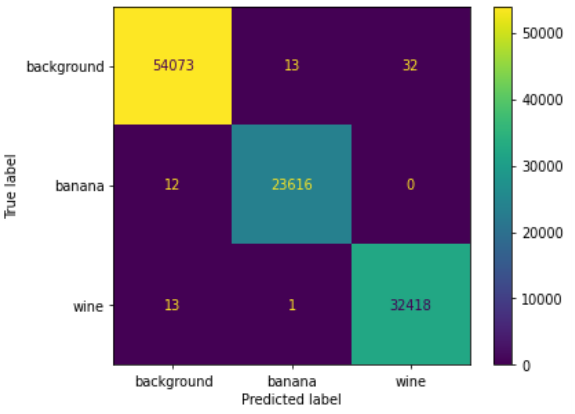

Figura 5: Matriz de confusión para Random Forest

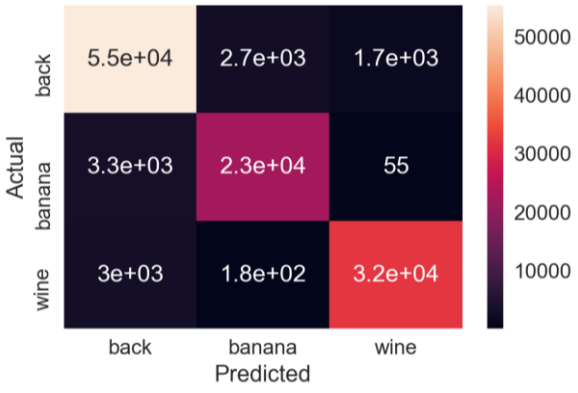

Figura 6: Matriz de confusión para Red Neuronal (NN)

Las siguientes gráficas muestran el efecto de no tener en cuenta los sensores de Humedad y Temperatura, así como usar hiperparámetros iniciales frente a los óptimos:

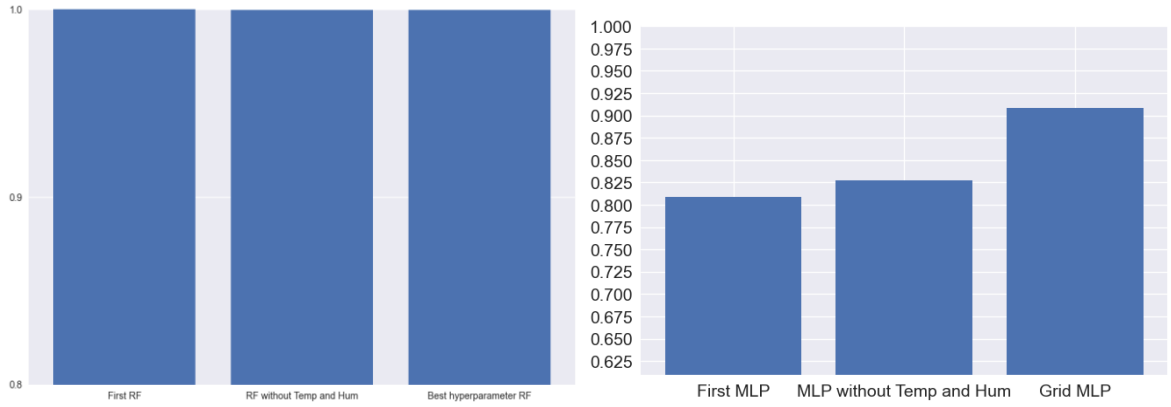

Figura 7: Comparativa RF-MLP con/sin Humedad y Temperatura, y con/sin parámetros óptimos

Las gráficas y estadísticas mostradas alumbran resultados prometedores: el rendimiento para las tres clases (*vino*, *banana*, y *background*) es suficientemente similar, con métricas de *accuracy* y *F1-score* muy superiores a las alcanzadas por el estado del arte [1]. Analizando las matrices de confusión de los modelos, podemos mencionar que sus distribuciones "en diagonal" indican que las tres clases están siendo clasificadas correctamente por lo general, con sólo unas pocas instancias que caen fuera de dicha diagonal. Es decir, las métricas estadísticas no sólo representan un buen rendimiento *en conjunto*, sino *particularmente* a cada clase.

Además de la Tabla 1, gráficas como las mostradas en la Figura 7 permiten ver que, comparativamente, el clasificador Random Forest es relativamente mejor que la Red Neuronal (NN-MLP). Tanto sus métricas, como su rendimiento con y sin los sensores de Humedad y Temperatura, anticipan una supremacía de este modelo. Además, su tiempo de clasificación es un poco más bajo (en torno a los 0.5 segundos), que el de la Red Neuronal, siendo esto muy importante para tomar una decisión entre uno y otro, dados los requerimientos de clasificación *online*.

Llegados a este punto, se tuvo curiosidad acerca de ver cómo respondía este modelo (Random Forest) frente a un conjunto de datos *completamente* nuevo. Si bien es cierto que, en validación simple, la proporción de datos de *test* nunca coincide con los de *train*, como ambas vienen del mismo dataset (ventanas de exposición con márgenes de 10 minutos), podríamos pensar que son en cierto modo "parecidas". Así pues, se creó un dataset de cero con la misma estrategia de atributos y muestras de los 5 minutos anteriores, pero *exceptuando* las ventanas de exposición a los estímulos. Es decir, se trata de un dataset íntegro de datos *background*.

El modelo *RandomForestClassifier* fue expuesto a este dataset, sin haber sido entrenado en él, y los resultados fueron los siguientes:

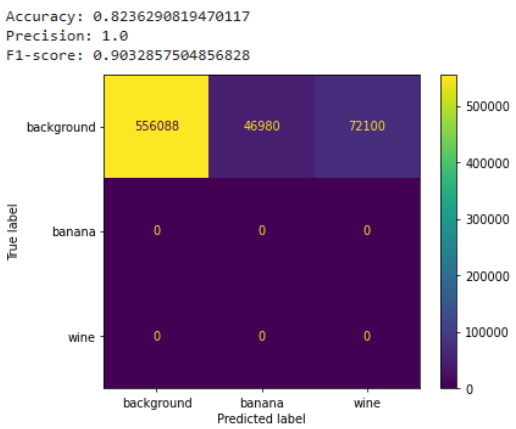

Figura 8: Rendimiento del modelo RF frente a un dataset desconocido de datos background.

Como puede observarse en la Figura 8, las métricas disminuyen un poco con respecto a las mostradas en la Tabla 1: Métricas de rendimiento para los modelos. La *F1-score*, más adecuada a entornos con desbalanceo de clases, pasa de un 99% a un 90.3%, y el *accuracy* de un 99% a un 82.3%. No obstante, estas disminuciones bien podrían respaldar nuestras conclusiones prometedoras acerca de este clasificador: incluso en su rendimiento más bajo, las métricas superan a las mostradas en el estado del arte [1], que recordemos, rondaban el 81%. Aparte, esta disminución en el rendimiento bien podría deberse al fenómeno de las "ventanas de exposición descentradas" que ya mencionamos en la Sección 2. Según esta teoría, algunos datos clasificados como *background* por encontrarse fuera de las ventanas marcadas por los metadatos, bien podrían ser realmente trazas de algún estímulo.

6 Conclusiones

El dataset original tiene ciertos inconvenientes, como las ventanas de exposición descentradas, la desproporción de datos *background* frente a los de *vino* y *banana*, o la ausencia de ciertos valores. Aparte, un estudio preliminar demostró que no era necesario tener en cuenta los sensores de Humedad y Temperatura para obtener buenas métricas de rendimiento. Los requerimientos de precisión y rapidez para adaptar el modelo a un entorno de clasificación *online*, motivaron la construcción de un dataset que tuviera en cuenta, para cada muestra, medidas estadísticas de los últimos 5 minutos. La consideración de márgenes de 10 minutos fuera de las ventanas de exposición mitigaba el fenómeno de las ventanas descentradas, así como aportar variedad a los datos clasificados como *background*.

Un estudio comparativo que hizo uso de tuneado de hiperparámetros, así como matrices de confusión, reveló que el clasificador Random Forest es notoriamente más efectivo que las Redes Neuronales. La simpleza de su diseño, unido a unas métricas de *accuracy* y *F1-score* cercanas al 99%, lo convierten en un buen candidato a funcionar en un entorno online (con tiempos de clasificación por muestra en torno a los 0.5 segundos).

El código y material adicional se puede encontrar en el repositorio [3].

Referencias

[1] Huerta, R., Mosqueiro, T., Fonollosa, J., Rulkov, N. F., & Rodriguez-Lujan, I. (2016). *Online decorrelation of humidity and temperature in chemical sensors for continuous monitoring*. Chemometrics and Intelligent Laboratory Systems, 157, 169-176.

[2] Pedregosa, F., Varoquaux, Ga"el, Gramfort, A., Michel, V., Thirion, B., Grisel, O., … others. (2011). *Scikit-learn: Machine learning in Python*. Journal of Machine Learning Research, 12(Oct), 2825–2830.

[3] Anivarro, C., Barahona, D., Cerrato, D. & Garitagoitia, D.T. (2023). *Exploración de modelos de aprendizaje automático para la clasificación online de mediciones de sensores MOX-Humedad-Temperatura*. Escuela Politécnica Superior, Universidad Autónoma de Madrid. https://github.com/danibt656/Gas-sensor-for-home-activity.

Nota de los autores: los anexos que vienen a continuación tienen una función meramente detallista sobre el contenido de las 8 páginas anteriores. El trabajo anterior es auto-explicativo y no exige la lectura de las siguientes páginas.

Anexo A: Medidas estadísticas de los sensores de Humedad y Temperatura

Las siguientes gráficas muestran la media (*mean*), mediana y desviación estándar (*dev*) para 3000 muestras de las tres clases existentes en el conjunto de datos (*vino*, *banana* y *background*).

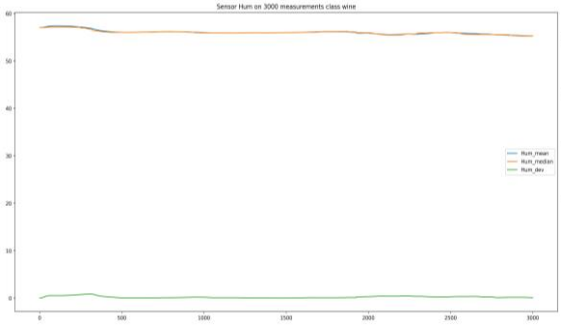

Figura 9: Sensor de Humedad, clase "vino"

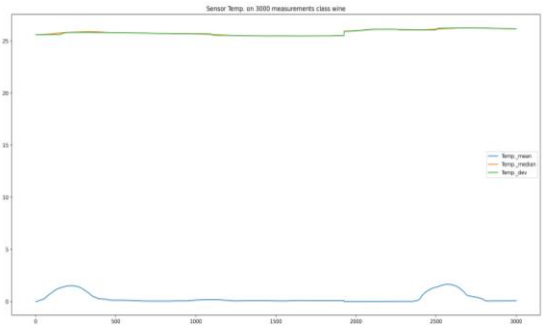

Figura 10: Sensor de Temperatura, clase "vino"

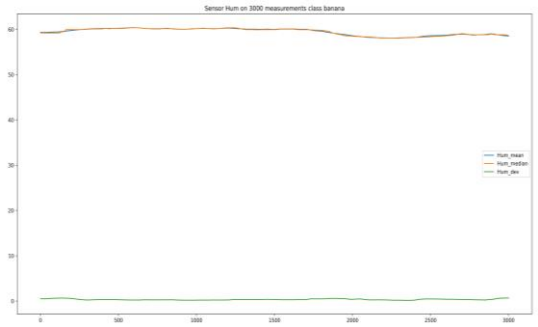

Figura 11: Sensor de Humedad, clase "banana"

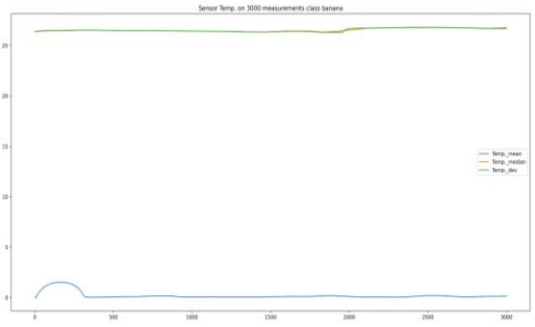

Figura 12: Sensor de Temperatura, clase "banana"

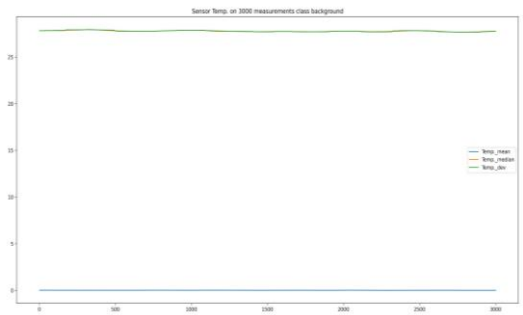

Figura 13: Sensor de Humedad, clase "background"

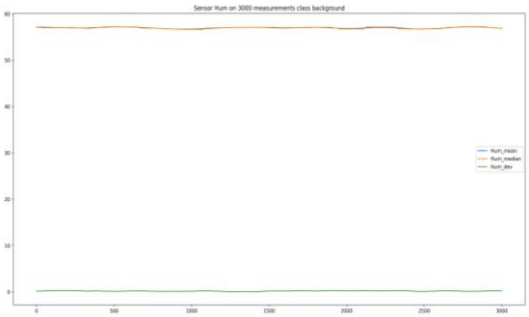

Figura 14: Sensor de Temperatura, clase "background"

Como se puede apreciar, las medidas de estos sensores presentan variaciones prácticamente imperceptibles para las tres clases, por lo que su uso se ha desestimado a la hora de construir el conjunto de datos de los experimentos.

Anexo B: Selección de hiperparámetros para los modelos

Los resultados mostrados en la Sección 5 corresponden a las versiones definitivas de los modelos de aprendizaje automático usados en los experimentos. Estas métricas fueron alcanzadas tras una fase de "tuneado" en la que se entrenaron y validaron diversas versiones de los modelos, modificando los valores de algunos hiperparámetros con el fin de buscar mejoras en las métricas.

Todos los resultados se han probado con validación simple, recogiendo las mismas métricas que las que muestra la Tabla 1. Se han considerado los siguientes hiperparámetros como los más relevantes de cada modelo:

RandomForestClassifier:

- `max_samples`: Decimal entre 0 y 1 que representa el porcentaje de muestras que se toman para hacer Bootstrap al entrenar los árboles. A probar entre 25%, 40%, 50%, 60%, 75% y 100% del número de muestras en el dataset.
- `max_features`: Número de atributos que se seleccionan aleatoriamente en cada punto de corte. Por defecto es la raíz cuadrada del número de atributos. Se probó de 1 a 8 atributos.
- `n_estimators`: Número de árboles (estimadores) en el bosque. Se probaron 3, 5, 7, 11, 21, 31, 51, 75 y 99 estimadores.

Para el clasificador Random Forest se probaron las distintas combinaciones, y la gráfica siguiente muestra una representación de *accuracy* y *F1-score* para todas ellas. El tamaño de los puntos es proporcional al tiempo de clasificación:

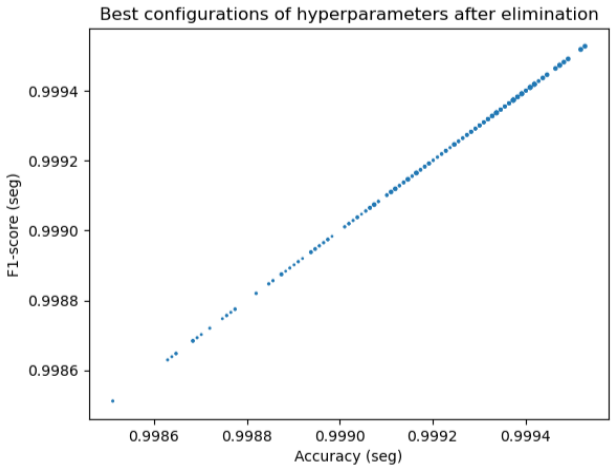

Figura 15: Tuneado de hiperparámetros para Random Forest

Obteniendo el mejor equilibrio entre métricas de rendimiento y tiempo de clasificación con `max_samples=1.0`, `max_features=8`, y `n_estimators=21`.

MLPClassifier:

- `hidden_layer_sizes`: Combinación de valores que contienen el número de capas ocultas y neuronas en dichas capas. Se acotaron las combinaciones probando varias que se ajustasen a las reglas típicas de estimación.
- `activation`: Función de activación de la capa oculta, se probó entre *tanh* y *ReLU* (devuelve f(x)=max(0,x)).
- `solver`: El optimizador de pesos, que compararemos entre *SGD* (descenso de gradiente estocástico) y *Adam* (se refiere a un optimizador estocástico basado en gradientes propuesto por Kingma, Diederik y Jimmy Ba).
- `learning_rate`: Tasa de aprendizaje empleada para la actualización de pesos, que compararemos entre la "constante" y "adaptativa" (en función de la *los function*), ambas inicializadas en 0.0001.

Se ha decidido emplear la *Grid Search* ofrecida por *Scikit-learn* para probar cuales son las mejores combinaciones acorde con su rank_test_score, de esta manera la búsqueda se hizo en el espacio de parámetros descritos anteriormente con validación cruzada de 3 *splits*.

La gráfica a continuación muestra como crece la función de mean_test_score a medida que nos acercamos al mejor resultado (Rank=1):

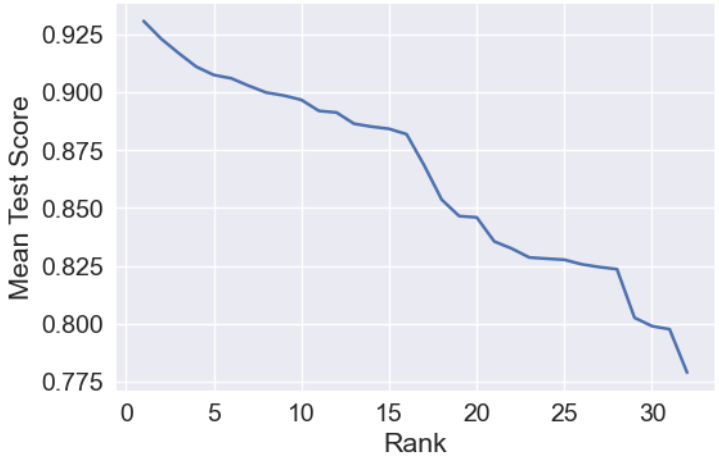

Figura 16: Tuneado de hiperparámetros para Red Neuronal

Siendo esta la mejor combinación: `hidden_layer_sizes=(24, 20, 16, 12)`, `activation='relu'`, `solver='sgd'` y `learning_rate='adaptive'`.