Escuela Técnica Superior de Ingeniería Informática

Máster Universitario en Ingeniería Informática

BitaHealth

Herramienta de detección de estrés mediante análisis de ECG con BiTalino

Realizado por

José María García Quijada

Sevilla, febrero 2025

Contenido

1. Introducción

El objetivo principal de este proyecto ha sido desarrollar un sistema capaz de determinar el estado emocional de un usuario (estrés, baseline, diversión, meditación, etc.) a partir de señales ECG adquiridas con un dispositivo **BiTalino**. Para ello, se ha llevado a cabo un proceso integral que incluye:

1. Recolección y preparación de datos (WESAD y muestras propias).
2. Extracción de características de **HRV** (Heart Rate Variability) con **NeuroKit2**.
3. Entrenamiento y validación de modelos de aprendizaje automático.
4. Implementación de un **servidor Flask** que aloja la IA entrenada.
5. Creación de una **interfaz web** con **React** para que el usuario final pueda subir su señal ECG y obtener un diagnóstico de forma amigable.

A lo largo de este documento se describirán todos los pasos llevados a cabo y las metodologías empleadas, buscando la transparencia y la reproducibilidad. Si bien existen numerosas investigaciones relacionadas con la detección de estrés a partir de señales fisiológicas, aquí se ha optado por un enfoque orientado a la practicidad y a la posibilidad de obtener resultados en entornos reales, valiéndose de un dispositivo económico como BiTalino.

2. Motivación y relevancia del proyecto

La detección del estrés a través de señales fisiológicas adquiere cada vez mayor importancia en el ámbito de la salud y el bienestar. El estrés crónico puede desencadenar múltiples patologías cardiovasculares y psicológicas. Desde una perspectiva académica, este proyecto ofrece:

- Una **visión práctica** de la aplicación de técnicas de procesado de señales y de aprendizaje automático.
- La **oportunidad** de integrar un flujo completo que va desde la toma de datos biomédicos hasta su interpretación y visualización final.
- Un caso de estudio **multidisciplinar**, combinando ingeniería biomédica, ciencia de datos y desarrollo web.

Además, se ha aprovechado un **dataset público reconocido**, WESAD, y se han realizado ensayos propios con datos reales capturados mediante electrodos para comprobar la generalización del modelo.

3. Objetivos

El objetivo principal es construir un sistema capaz de **clasificar** si una persona se encuentra en estado de estrés (o en otras categorías como baseline, diversión o meditación) basándose en datos de ECG.

Además, también se cuenta con **objetivos específicos** como:

- Automatizar la **extracción de características HRV** mediante el uso de librerías como *NeuroKit2*.

- Implementar un **pipeline de aprendizaje automático** que incluya distintas etapas de preprocesado (imputación, escalado, reducción de dimensionalidad, balance de clases).
- Evaluar diversos **modelos** (k-NN, Decision Tree, Random Forest, SVM, MLP, etc.) e identificar aquel con mejor desempeño.
- **Desplegar** el modelo seleccionado en un servidor Flask que provea un endpoint para la inferencia en tiempo casi real.
- Crear una **interfaz web** (en React) para que el usuario final pueda subir archivos .txt de ECG de BiTalino y visualizar el resultado del análisis.

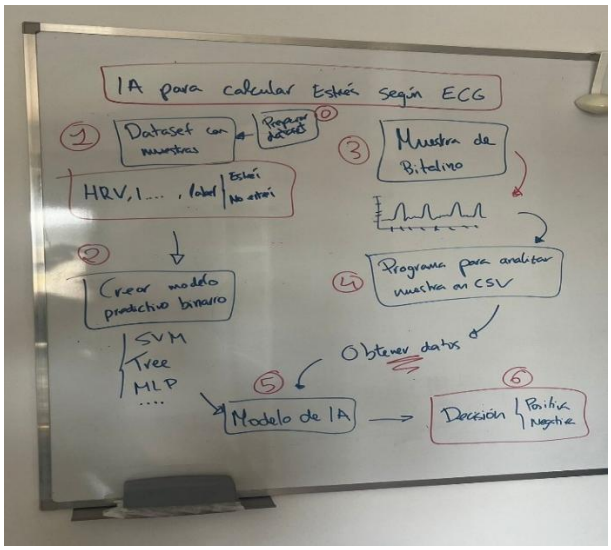

Ilustración 1: Diagrama de flujo propuesto el día 1

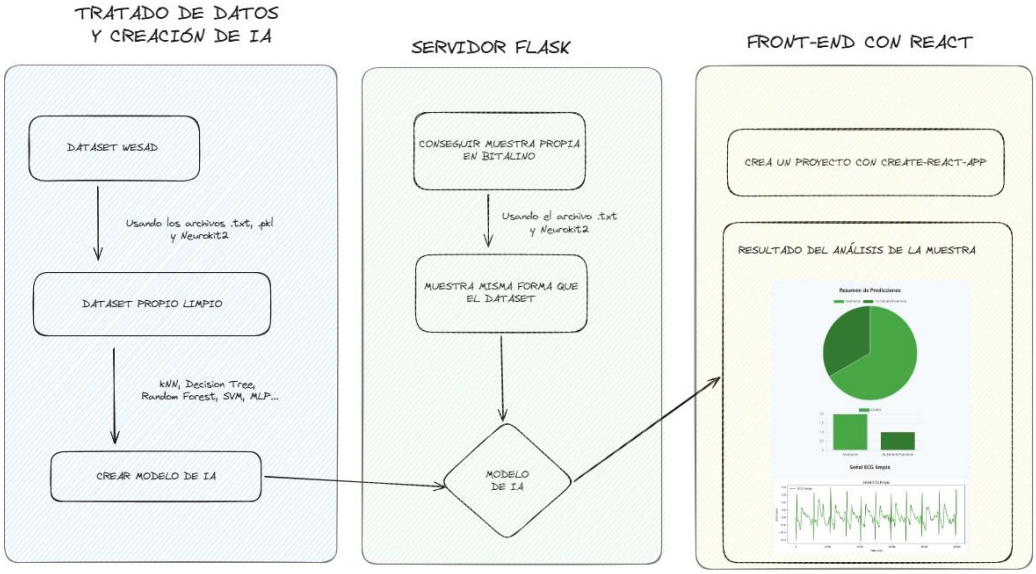

Ilustración 2: Diagrama de flujo final

4. Herramientas y entorno de desarrollo

- **Lenguajes de programación**:
  - Python (para procesamiento de datos, entrenamiento del modelo e implementación del servidor Flask).
  - JavaScript (React para el Frontend).
- **Bibliotecas y frameworks principales**:
  - *Flask* (conexión API y despliegue de la IA).
  - *NeuroKit2* (procesamiento de señales ECG, extracción de HRV).
  - *Scikit-learn* y *Imbalanced-learn* (entrenamiento y evaluación de modelos, oversampling, etc.).
  - *Pandas* y *NumPy* (manipulación de datos).
  - *Matplotlib* y *Seaborn* (visualización).
  - *React* (frontend).
  - *create-react-app* (estructura base del proyecto en React).
- **Hardware**:
  - Dispositivo **BiTalino** (para captura de señales fisiológicas, en particular ECG).
- **Dataset principal**:
  - **WESAD**, que incluye grabaciones de varios sujetos y ofrece, entre otros, archivos .txt con la señal cruda de ECG y archivos .pkl con las etiquetas originales del experimento.

El proyecto se desarrolló en un entorno local, usando un sistema operativo Windows (aunque el proceso es fácilmente replicable en otros entornos con Python y NodeJS instalados).

5. Descripción del dispositivo BiTalino y análisis de mercado

En esta sección se expone detalladamente el funcionamiento del **dispositivo BiTalino**, sus características técnicas y las razones de su elección para este proyecto. Dado que el **RespiBAN**, desarrollado por el mismo fabricante, también ha sido empleado para la adquisición de datos (en el dataset WESAD), se incluirá un breve apartado con sus especificaciones y similitudes respecto a BiTalino. Finalmente, se realiza una comparativa con otros sistemas similares en el mercado, detallando ventajas, limitaciones y rango de precios.

5.1. Dispositivo BiTalino

BiTalino es un kit de desarrollo orientado a la captura de **señales biomédicas** y al prototipado de aplicaciones en el ámbito de la salud. Fue diseñado por PLUX Wireless Biosignals, empresa especializada en plataformas de monitorización fisiológica para investigación y desarrollo.

5.1.1. Componentes principales

- **Módulos de captura**:

  BiTalino se distribuye en varios modelos o kits (por ejemplo, *bitalino (r)evolution*) que incluyen sensores para:

  - ECG (Electrocardiografía)
  - EMG (Electromiografía)
  - EDA (Electrodermal Activity)
  - Acc (Acelerómetro)
  - LUX (Sensor de luz)
  - TMP (Temperatura), entre otros.

- **Placa base**:

  Integra un microcontrolador (en la versión (r)evolution, un PIC24FJ64GA o similar), conversores analógico-digitales (ADC) de 10 a 12 bits según la versión, y un módulo de comunicación (generalmente Bluetooth o USB).

- **Frecuencia de muestreo**:

  Oscila entre 100 y 1000 Hz para los distintos canales, configurable según las necesidades del usuario.

- **Alimentación**:

  Usualmente funciona con baterías recargables, aunque también puede operar conectada a un ordenador por USB.

5.1.2. Características técnicas del canal ECG

1. **Resolución**:
   El ADC de BiTalino ofrece típicamente 10 bits en sus primeras versiones, y hasta 12 o 16 bits en las más recientes. Esto repercute en la resolución de la señal ECG (por ejemplo, ±1,5 mV).

2. **Rango de medición**:

   Por lo general, BiTalino amplifica la señal de ECG para situarla en un rango de 0-3,3 V (o 0-5 V), con un offset intermedio para poder capturar voltajes positivos y negativos.

3. **Facilidad de uso**:

   Los electrodos se conectan mediante cables específicos, y el kit normalmente incluye parches de un solo uso para colocar en el pecho del usuario o en las extremidades.

5.1.3. Ventajas de BiTalino

- **Modularidad**: Se pueden adquirir solamente los sensores necesarios.
- **Bajo coste** comparado con sistemas médicos de alta gama.

- **Compatibilidad** con múltiples entornos (Python, C++, Android, etc.) y ejemplos de código que facilitan el desarrollo rápido de prototipos.

5.2. Dispositivo RespiBAN

RespiBAN es otro dispositivo de la familia PLUX (mismos fabricantes de BiTalino) que también está concebido para la medición de **señales respiratorias y cardiacas**, entre otras. Dentro del dataset WESAD, RespiBAN fue empleado para registrar ECG, EDA y respiración, entre otras señales.

5.2.1. Similitudes con BiTalino

- **Origen**: Ambos provienen de PLUX, por lo que comparten características propias de la empresa y se enfocan en aplicaciones de telemetría biomédica e I+D.
- **Captura de ECG**: RespiBAN, al igual que BiTalino, ofrece un canal de electrocardiografía con resolución y frecuencia de muestreo adecuadas para la mayoría de aplicaciones de investigación.
- **Software y documentación**: El fabricante proporciona documentación técnica y ejemplos de uso similares para BiTalino y RespiBAN.

5.2.2. Diferencias destacables

En cuanto al enfoque principal, BiTalino se percibe como un *kit educativo / de prototipado* modular, en el que el usuario puede intercambiar sensores. En cambio, RespiBAN se dirige a entornos donde se precise un dispositivo más específico para monitorización de la respiración y el ECG, con la posibilidad de incorporar cintas torácicas o bandas pectorales.

El formato y ergonomía, ya que BiTalino se suele presentar como una placa de desarrollo + sensores y RespiBAN puede presentarse en una carcasa cerrada, más cómoda de llevar, e incluye a menudo correas diseñadas para la medición continua en el torso.

Finalmente, en el coste, debido a que RespiBAN puede tener un precio elevado, sobre 900€, si se adquiere con todos los accesorios de monitoreo de respiración y ECG. BiTalino, por su parte, tiene distintos *kits* que oscilan en una gama de precios entre 100€ y 300€, dependiendo del número de sensores incluidos.

En el proyecto descrito, los archivos .txt y .pkl del dataset WESAD corresponden a registros tomados con **RespiBAN**, lo que explica la coincidencia de formatos y la interfaz similar a la de BiTalino.

5.3. Otros dispositivos ECG en el mercado

Existen múltiples sistemas comerciales y de investigación orientados a la obtención de señales fisiológicas (especialmente ECG). A continuación, se nombran algunos ejemplos que pueden encontrarse con relativa facilidad:

1. **Shimmer3 ECG Unit**
    - **Características**: Dispositivo portátil para ECG, EMG, GSR. Frecuencias de muestreo de hasta 1 kHz, conectividad Bluetooth.

- **Precio aproximado:** Entre 400€ y 1000€, dependiendo de la configuración.
- **Ventajas:** Robusto en entornos de investigación, buen soporte de software.
- **Limitaciones:** Coste elevado, menos orientado a la educación.

2. **e-Health Sensor Platform (Libelium / Cooking Hacks)**
   - **Características:** Placa Arduino/Raspberry con módulos para ECG, EDA, pulsioxímetro, etc.
   - **Precio aproximado:** 200€ a 300€.
   - **Ventajas:** Integración sencilla con entornos Maker (Arduino IDE), comunidad activa.
   - **Limitaciones:** Focalizada en prototipado, requiere cierta habilidad con Arduino y soldadura para un montaje fiable.
3. **Holter ECG convencionales**
   - **Características:** Equipos médicos certificados para monitorización continua de 24-48 horas en pacientes cardíacos.
   - **Precio aproximado:** Desde 800€ a varios miles de euros, según modelo y fabricante.
   - **Ventajas:** Validación clínica y precisión elevada, software específico para cardiología.
   - **Limitaciones:** Son dispositivos pensados para diagnóstico médico, con menor flexibilidad para manipulación de datos brutos o prototipado.

5.4. Comparativa y justificación de la elección

En la siguiente tabla se resume una **comparación** aproximada entre los principales dispositivos mencionados, centrándonos en **coste**, **modularidad**, **soporte de software** y **orientación**:

Dispositivo	Precio aprox	Modularidad	Soporte	Orientación
BiTalino	100 – 300 €	Alta	Alto	Académica I+D
RespiBan	900 €	Media	Alto	I+D e investigación
Shimmer3	400 – 1000 €	Media	Bueno (SDK propio)	Investigación
eHealth	200-300€	Media - Alta	Bueno (Arduino)	Prototipado
Holter	800 €	Baja	Cerrado	Clínico

5.4.1 Razones para la elección de BiTalino / RespiBAN

- **Integración fácil** con entornos de programación (Python, C++), facilitando la extracción y análisis de HRV.
- **Amplia documentación** y comunidad, especialmente BiTalino, que cuenta con numerosos ejemplos de código y foros de usuarios.

- **Precio razonable** en comparación con sistemas médicos de grado clínico o con plataformas como Holter, la inversión es menor.
- **Confiabilidad suficiente** para proyectos de investigación no clínicos que requieran señales de ECG, EDA o EMG de calidad aceptable.

Dado que el **dataset WESAD** emplea RespiBAN (un dispositivo de la misma familia), se ha beneficiado de un **formato de archivo** muy similar al de BiTalino, lo cual simplifica el preprocesado y la extracción de señales para este proyecto.

6. Preparación y procesamiento de datos

6.1 Estructura de los datos originales

En el dataset **WESAD**, cada sujeto (S1, S2, S3...) cuenta con:

- Un archivo .txt (por ejemplo, S1_respiban.txt) que contiene múltiples columnas de señales (ECG, EDA, EMG, etc.).
- Un archivo .pkl con las etiquetas asociadas a cada ventana de tiempo (por ejemplo, estrés, diversión, baseline, etc.).

El formato del .txt de BiTalino incluye un **encabezado** con metadatos en formato JSON y después columnas tabuladas. Un ejemplo de línea de datos es:

```
# OpenSignals Text File Format
# {"00:07:80:D8:AB:58": {"sensor": ["ECG", "EDA", "EMG", "TEMP", "XYZ", "XYZ", "XYZ", "RESPIRATION"],
#"device name": "RespiBan", "column": ["nSeq", "DI", "CH1", "CH2", "CH3", "CH4", "CH5", "CH6", "CH7", "CH8"],
#"sync interval": 2, "time": "9:39:1.0", "comments": "", "device connection": "BTH00:07:80:D8:AB:58",
#"channels": [1, 2, 3, 4, 5, 6, 7, 8], "date": "2017-5-22", "mode": 0, "digital IO": [0, 1],
#"firmware version": 770, "device": "biosignalsplux", "position": 0, "sampling rate": 700,
#"label": ["CH1", "CH2", "CH3", "CH4", "CH5", "CH6", "CH7", "CH8"], "resolution": [16, 16, 16, 16, 16, 16, 16, 16],
#"special": [{}, {}, {}, {}, {}, {}, {}, {}]}}
# EndOfHeader
0   0   31053   14694   32651   29553   37415   32324   31659   31364
1   0   30861   14680   32983   29556   37417   32333   31663   31379
2   0   30644   14684   32823   29561   37409   32340   31655   31367
3   0   30374   14701   32569   29553   37405   32341   31671   31076
4   0   30103   14684   32794   29561   37409   32342   31681   31365
5   0   29839   14753   32929   29567   37414   32334   31703   31361
```

Ilustración 3: Archivo .txt del dataset WESAD

De todas esas columnas, sólo nos interesaba la correspondiente a la señal ECG cruda para el posterior análisis de variabilidad cardiaca.

Asimismo, el archivo .pkl contenía las **etiquetas** para cada ventana, indicándonos la clase (0 = No definido/Transitorio, 1 = Baseline, 2 = Estrés, 3 = Diversión, 4 = Meditación, etc.).

6.2 Script de extracción de características (HRV)

Para unificar y automatizar este proceso, se desarrolló un **script en Python** que:

1. Lee el .txt.
2. Se queda únicamente con la columna del ECG y la normaliza o transforma a unidades físicas (mV).
3. Lee el archivo .pkl para recuperar las etiquetas.

4. Segmenta la señal en ventanas de **30 segundos** (adaptable) según la **frecuencia de muestreo** de BiTalino (en este caso, 700 Hz).
5. Aplica **NeuroKit2** para:
    - ❖ Limpiar la señal ECG (filtrado de ruido).
    - ❖ Detectar picos R.
    - ❖ Calcular las métricas de **HRV** en el dominio del tiempo y de la frecuencia (p. ej., HRV_MeanNN, HRV_SDNN, HRV_RMSSD, HRV_HF, HRV_LF, etc.).
6. Almacena todas las características en un **archivo CSV**.

Gracias a este proceso se obtuvieron varios ficheros CSV con estructura similar:

Ilustración 4: Dataset propio con datos refinados

Estos CSV finales representaron la **base de datos** con la cual se entrenó la IA.

7. Entrenamiento de la IA

7.1 Creación de diferentes versiones del dataset

Para robustecer el análisis y asegurarnos de que el modelo pudiera generalizar, se diseñó un **pipeline de preprocesamiento** con distintas aproximaciones, generando cuatro versiones del mismo dataset:

1. **Dataset 1**:
    - ❖ Eliminación directa de filas con valores nulos.
    - ❖ *LabelEncoder* para la variable objetivo.
    - ❖ *StandardScaler* para escalar las columnas numéricas.
2. **Dataset 2**:
    - ❖ Imputación de valores numéricos mediante la mediana y de valores categóricos por la moda.
    - ❖ *LabelEncoder* y *StandardScaler* de igual modo.
3. **Dataset 3**:
    - ❖ Eliminación de columnas con más del 30% de datos nulos.
    - ❖ Posterior imputación y escalado.

4. **Dataset 4**:
    - ❖ Imputación general (mediana para numéricos, moda para categóricos).
    - ❖ *LabelEncoder*, *StandardScaler*.
    - ❖ **PCA** para reducción de dimensionalidad (eligiendo componentes que expliquen el 95% de la varianza).
    - ❖ **SMOTE** para balancear las clases en caso de que existiese sesgo o desequilibrio.

7.2 Entrenamiento y validación de diferentes modelos

Se codificó un **script de entrenamiento** que, para cada una de las cuatro versiones de dataset, probaba los siguientes algoritmos:

- ❖ **k-NN** (K Nearest Neighbors): Método basado en instancias, sencillo y efectivo en muchos problemas de clasificación.
- ❖ **Árbol de Decisión** (Decision Tree): Modelo interpretativo, útil para explorar la importancia de variables.
- ❖ **Random Forest**: Ensamble de múltiples árboles de decisión, con alta capacidad de generalización.
- ❖ **Gradient Boosting**: Potente metodología basada en aditividad de modelos débiles de forma secuencial.
- ❖ **Regresión Logística**: Modelo estadístico lineal, interpretable y eficiente.
- ❖ **SVM** (Support Vector Machine): Método robusto para problemas lineales y no lineales (mediante kernels).
- ❖ **MLP Classifier** (Multilayer Perceptron): Aproximación de red neuronal sencilla, con una o varias capas ocultas.

Se aplicó **GridSearchCV** (con validación cruzada) para encontrar los **hiperparámetros óptimos** de cada modelo. Tras el entrenamiento, se evaluaban métricas como **Accuracy**, **Precision**, **Recall**, **F1** y **AUC**. Finalmente, se almacenaban los resultados en CSV y se generaban gráficos de barras para compararlos.

7.3 Ajuste de hiperparámetros y métricas

Se definieron **celdas** de hiperparámetros para cada modelo. Por ejemplo, para k-NN se exploraron valores de n_neighbors (1, 3, 5, 7), tipo de distancia, etc. Para Random Forest, se variaron n_estimators, max_depth, min_samples_split, etc.

Después del entrenamiento, se compararon las métricas:

- ❖ **Accuracy** (proporción de aciertos global).
- ❖ **Precisión** (precision, macro).
- ❖ **Recall** (sensibilidad, macro).
- ❖ **F1-score** (promedio armónico de precisión y recall, macro).
- ❖ **AUC** (área bajo la curva ROC, calculada en modo multiclase con One-vs-Rest).

Los resultados finales se almacenaron en CSV y se generaron **gráficos de barras** para poder visualizar qué combinación de dataset y modelo resultaba más favorable.

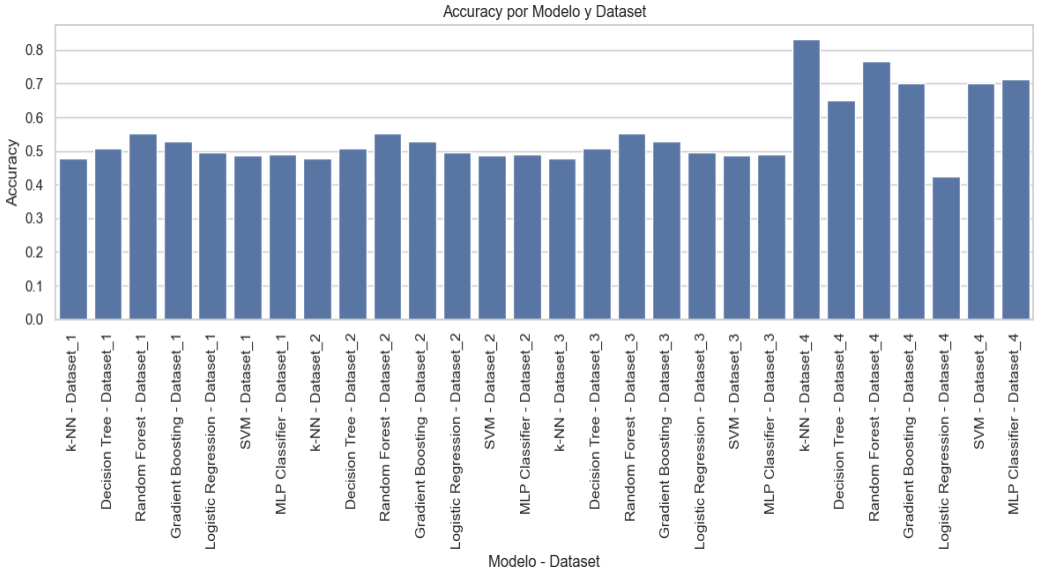

Ilustración 5: Precisión del modelo por datasets

7.4 Resultados obtenidos

Al observar los **tres primeros** datasets (1, 2 y 3), los rendimientos oscilaron en torno al 50-55% de accuracy en el mejor de los casos, indicando que el problema tenía cierta complejidad y que posiblemente habría una falta de discriminación en los datos crudos o un desequilibrio de las clases.

En cambio, el **Dataset 4** (el que incluye PCA y SMOTE) mostró mejoras notorias. El **modelo k-NN** en ese dataset alcanzó alrededor de un **83% de exactitud** y un **AUC > 0.88**, superando a otros algoritmos probados. Random Forest y Gradient Boosting tuvieron también métricas respetables (70-76% de accuracy), pero se seleccionó k-NN por su mayor precisión global y su robustez en validación cruzada.

A modo de ejemplo, algunos resultados del *Dataset 4*:

Modelo	Accuracy	F1-Score	AUC
K-NN	0.8311	0.8120	0.8944
Random Forest	0.7669	0.7525	0.9459
Gradient Boosting	0.7027	0.6958	0.9027
MLP Classifier	0.7134	0.6999	0.8899

Dada esta comparativa, se optó por **k-NN con Dataset 4** como **solución final**.

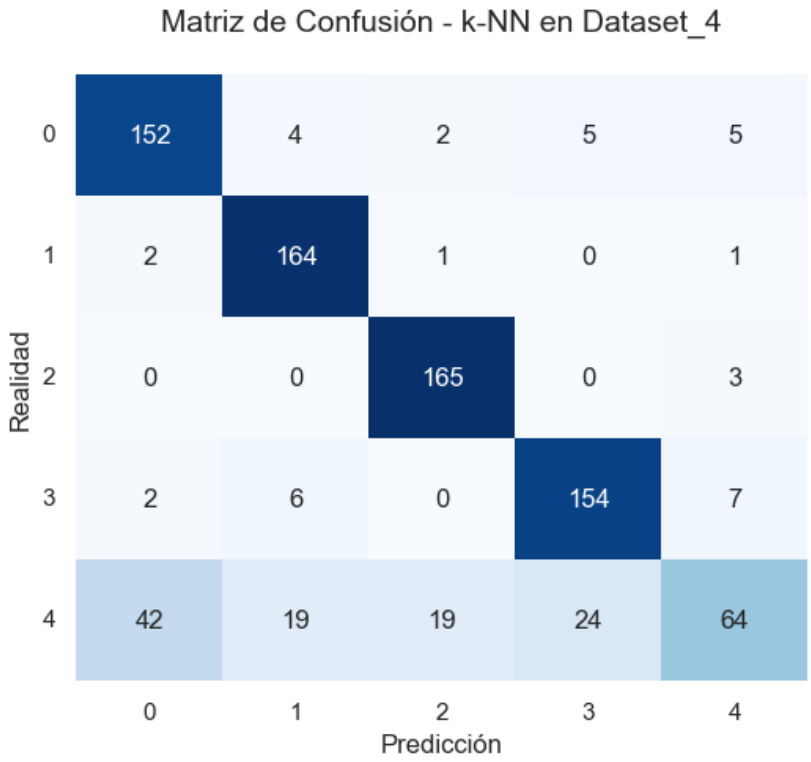

Ilustración 6: Matriz de confusión de kNN

7.5 Exportación y guardado de artefactos

Para la puesta en producción del modelo, se guardaron en disco:

- El **modelo entrenado** en formato .pkl.
- Los **objetos de preprocesado** (imputador, scaler, PCA, SMOTE, LabelEncoder), necesarios para replicar en tiempo de inferencia los mismos pasos aplicados en entrenamiento.

8. Implementación del servicio con Flask

Para ofrecer la IA de forma centralizada, se creó un **servidor Flask** que expone un endpoint "/upload":

1. Recibe el archivo .txt del dispositivo BiTalino.
2. Lo procesa (lee y limpia la señal ECG, segmenta en ventanas, extrae métricas HRV con **NeuroKit2**).
3. Aplica los pasos de preprocesado (imputación, escalado, PCA) tal y como se hizo en entrenamiento.
4. Carga el modelo k-NN en formato .pkl y obtiene la **predicción** para cada ventana (por ejemplo: estrés, meditación, baseline, no definido…).
5. Devuelve un **JSON** con las predicciones y un pequeño **resumen** con el conteo y el porcentaje de cada clase.

Asimismo, el servidor genera algunos **gráficos** (señal ECG limpia, intervalos RR, histograma, etc.) en base64 y los incluye en la respuesta para que el frontend pueda mostrarlos.

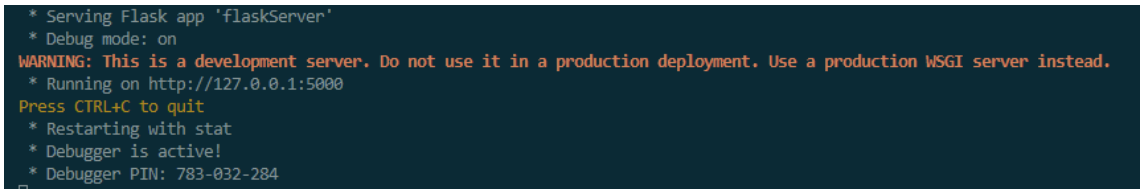

Ilustración 7: Servidor flask funcionando

8.1 Estructura del código Flask

- **Rutas principales**:
  - "/upload": Método POST que recibe el archivo .txt.
- **Funciones** de utilidad:
  - process_bitalino_txt(): para segmentar la señal e ir extrayendo características en ventanas de 10 o 30 segundos.
  - run_model(): para cargar el pipeline de preprocesado y el modelo k-NN, realizar la inferencia y devolver los resultados.
  - generate_graphs(): para generar las visualizaciones (ECG, Poincaré Plot, FFT, etc.) y retornarlas como strings base64.

Esta arquitectura modular facilita la mantenibilidad y escalabilidad del proyecto.

9. Interfaz de usuario con React

La última fase consistió en construir una **aplicación web** con **React** que brindase una experiencia intuitiva.

9.1 Objetivos de la interfaz

La aplicación web en **React** fue concebida para:

- Permitir **cargar** (mediante un botón o drag-and-drop) un archivo .txt proveniente de BiTalino.
- Mostrar **retroalimentación** en tiempo real sobre el estado de la solicitud ("Procesando…", "Completado", etc.).
- Reflejar los **resultados** en forma de:
  - Listado de predicciones ventana a ventana.
  - Gráficos de torta (Pie Chart) y de barras (Bar Chart) con la distribución de etiquetas.
  - Imagen base64 de la señal ECG limpia, sus picos R y otras visualizaciones (Poincaré, FFT, histograma, etc.).
- Permitir **exportar** todo el informe a PDF, usando librerías como html2canvas y jspdf.

9.2. Implementación del componente principal

Se creó un archivo FileUploadComponent.js, en el que se definen:

- **Estados**: txtFile, result, loading, etc.
- **Funciones** para manejar el evento de selección de archivo y el envío al servidor Flask mediante fetch.
- **Sección de resultados**: que parsea el JSON recibido y lo muestra en la interfaz.
- **Funcionalidad** para la **descarga del PDF**, capturando la sección del DOM que contiene gráficos y texto.

El uso de **React** permite una organización clara de la lógica y la posibilidad de reutilizar componentes en futuros desarrollos. Asimismo, se integró **Chart.js** para la representación de gráficos en forma nativa dentro de la SPA (Single-Page Application).

Ilustración 8: Front-end React

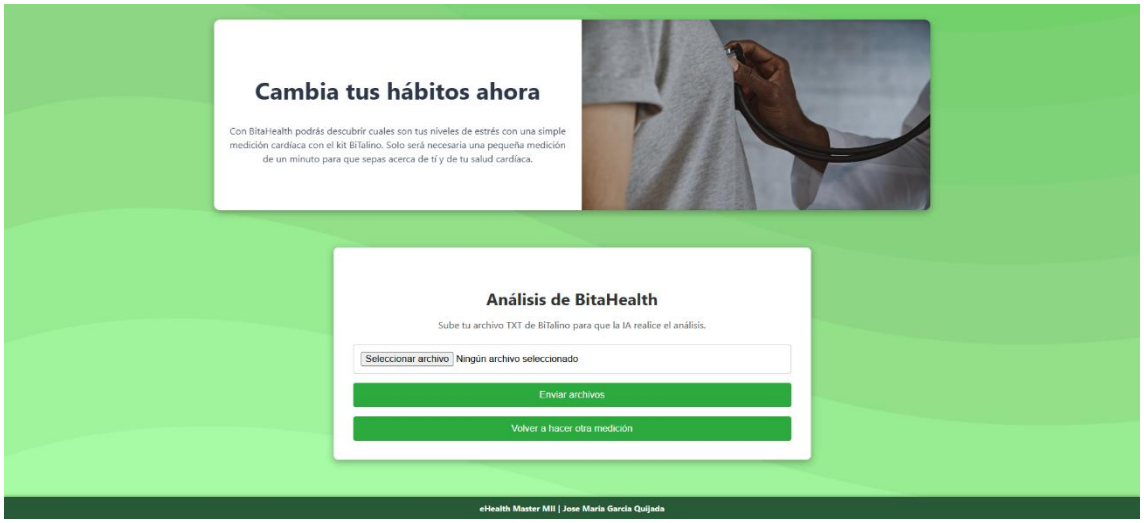

Ilustración 9: Front-end React

10. Resultados en pruebas reales

Al probar el sistema con una **muestra real** capturada con BiTalino (usando los electrodos en el pecho):

- Se recogieron aproximadamente 30 segundos de señal y se generaron 3 ventanas.
- Las predicciones finales fueron, por ejemplo, 2 ventanas clasificadas como "Meditación" y 1 ventana como "No Definido/Transitorio".
- La interfaz React interpretaría un JSON como este:


```
{
  "predictions": [
    "Meditación",
    "Meditación",
    "No Definido/Transitorio"
  ],
  "summary": {
    "Meditación": {
      "count": 2,
      "percentage": 66.67
    },
    "No Definido/Transitorio": {
      "count": 1,
      "percentage": 33.33
    }
  },
  "total_windows": 3
}
```


Ilustración 10: JSON que recibe la web desde el servidor

Estos resultados se reflejaron correctamente en las gráficas de la web y en un PDF exportable.

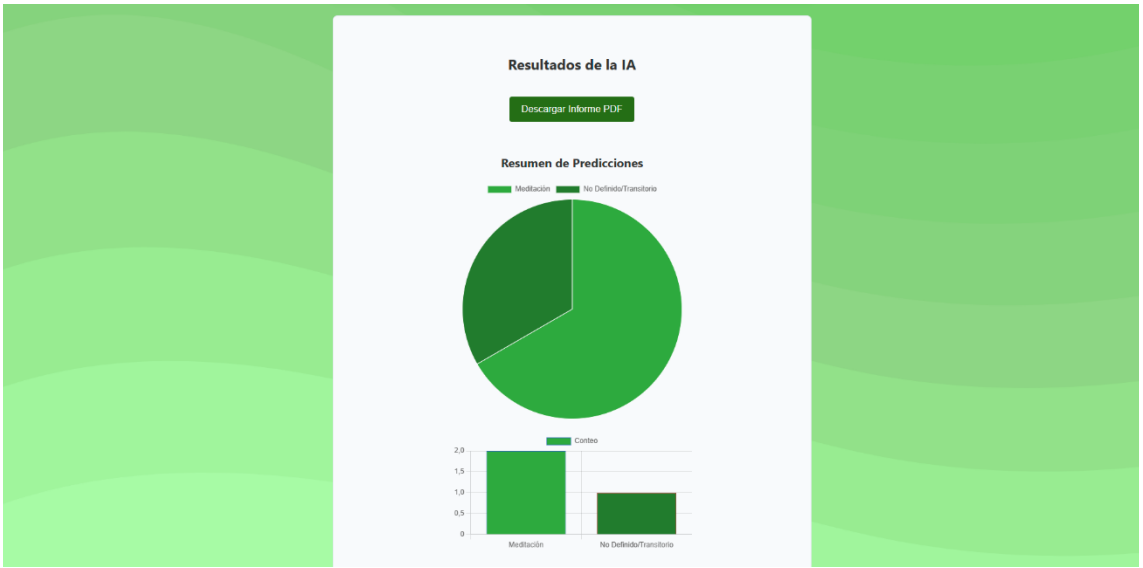

Ilustración 11: Resultados Front-end React

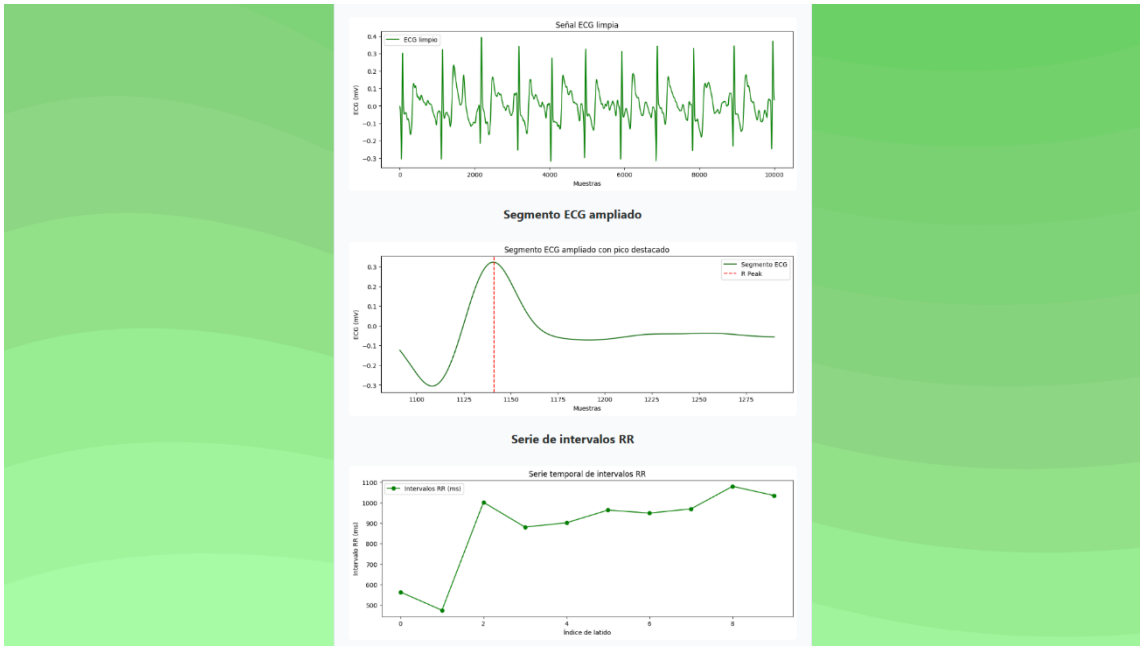

Ilustración 12: Resultados Front-end React

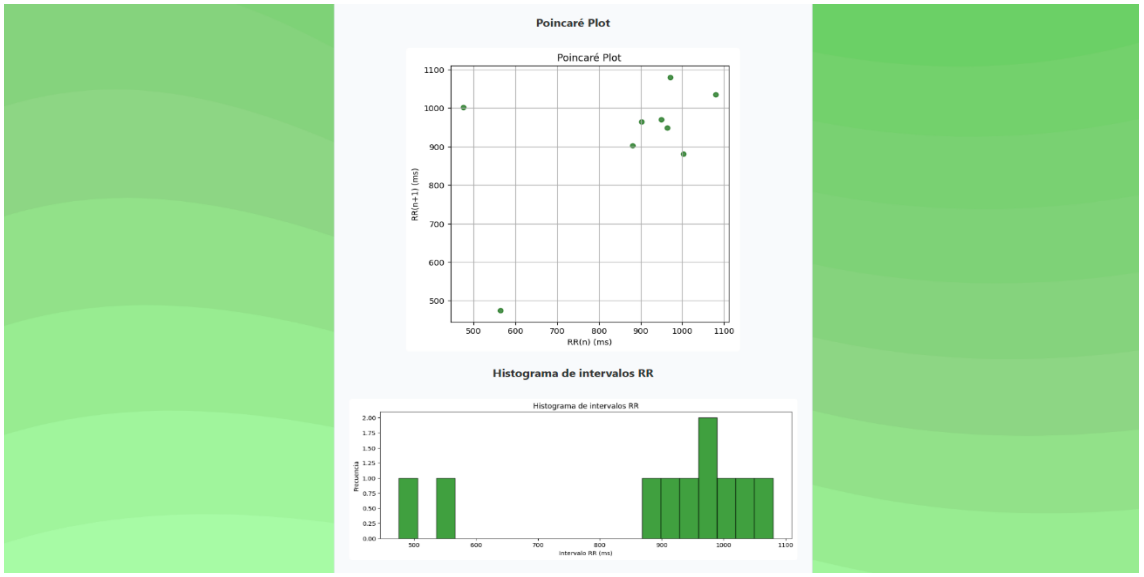

Ilustración 13: Resultados Front-end React

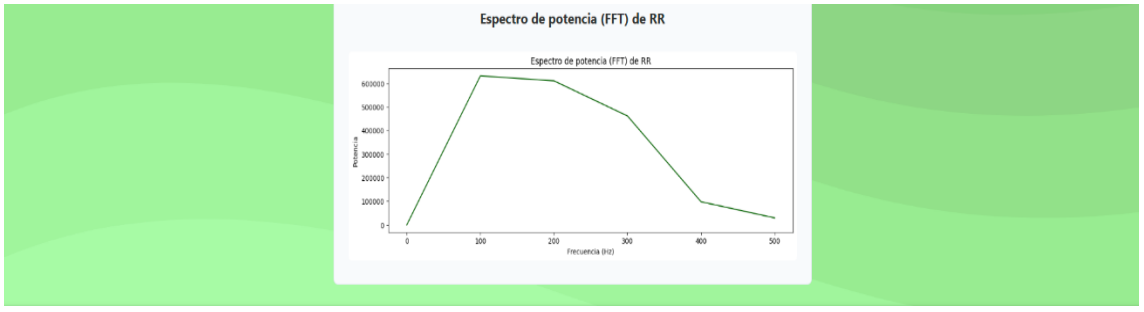

Ilustración 14: Resultados Front-end React

11. Informes técnicos de certificación (Marcado CE)

En esta sección se plantea la perspectiva de que **BitaHealth**, en tanto producto final, busca certificar su uso en el ámbito de la Unión Europea. La herramienta combina **BiTalino** para capturar en tiempo real las señales fisiológicas del usuario y, además, utiliza un **modelo de IA** entrenado, en parte, con datos de **RespiBAN**. Analizamos el marco normativo y los requisitos que deberían considerarse para lograr el **Marcado CE** como dispositivo sanitario.

11.1. Directivas y reglamentos aplicables y requisitos

Con la plena entrada en vigor del **Reglamento (UE) 2017/745** (Medical Device Regulation o MDR), cualquier dispositivo que pretenda tener una **función sanitaria** o **diagnóstica** se encuentra bajo su ámbito de aplicación. Si BitaHealth se utiliza exclusivamente para fines de bienestar o investigación (por ejemplo, detección de estrés de carácter no médico), podría intentar situarse fuera del MDR y actuar como herramienta de investigación.

Sin embargo, por el contrario, si se comercializa para **monitorear** o **detectar** condiciones que pudiesen considerarse patológicas (por ejemplo, estrés crónico con repercusión clínica, alteraciones del ritmo cardiaco, etc.), se aplicaría esta normativa.

En tal caso, que es el que realmente queremos para poder exprimir esta sección, se deben cumplir los requisitos generales de seguridad y funcionamiento (Essential Requirements) descritos en el Anexo I del MDR, que abarcan:

- **Seguridad eléctrica** y compatibilidad electromagnética (ej., siguiendo EN 60601-1, EN 60601-1-2).
- **Gestión de riesgos** (ISO 14971) y **evaluación de usabilidad**.
- **Demostración de exactitud** y fiabilidad de las mediciones cardiacas (podrían apoyarse en ensayos comparativos con equipos de referencia).
- **Documentación de software** (validaciones, pruebas unitarias, ciclo de vida del software de la IA, etc.).

Debido a que BitaHealth hace uso de **IA** entrenada con datos RespiBAN y opera con un **dispositivo físico BiTalino**, parte del reto es demostrar que ambos subsistemas (hardware + software) funcionan de manera segura y cumplen con la finalidad declarada.

11.2. Clasificación del producto sanitario

El MDR define reglas de clasificación basadas en **riesgo**. Para un dispositivo que **monitoriza** señales como ECG y EDA, y que potencialmente **ayuda** en la detección de estrés u otras alteraciones fisiológicas, a menudo se situaría en **Clase IIa** (por ejemplo, Regla 10 u 11).

BiTalino se vende habitualmente como un **kit de prototipado** y herramienta educativa, por lo que, en principio, **no** se considera un producto sanitario final.

RespiBAN puede también presentarse como un dispositivo de investigación / I+D.

No obstante, **BitaHealth**, al unir estas tecnologías y proveer una **finalidad de monitorización de parámetros** con un posible impacto en el diagnóstico/pre-diagnóstico (p. ej., estrés crónico),

puede requerir clasificación en **Clase IIa** si se declara un uso médico o preventivo. En esta clase, se exige la intervención de un Organismo Notificado para la evaluación de conformidad.

11.3. Documentación técnica elaborada

Para obtener el Marcado CE, BitaHealth debería preparar un **Expediente Técnico** (Technical Documentation) que incluyera, entre otros, los siguientes apartados:

1. **Descripción del producto y finalidad de uso**

Explicar el hardware BiTalino y cómo se integra con la IA. Aclarar si se limita a la detección de estrés en entornos no clínicos o si se persigue uso médico (por ejemplo, detección de arritmias leves). Describir la parte de datos provenientes de RespiBAN para el entrenamiento y cómo se mantiene la trazabilidad de esa información (protección de datos, etc.).

2. **Gestión de riesgos (ISO 14971)**

Identificar los posibles peligros (electrodos en contacto con el cuerpo, falsas alarmas de estrés, interferencias electromagnéticas, etc.). Asegurar la calidad de las mediciones, la calibración del ECG, la validez del algoritmo de IA, etc.

3. **Evaluación clínica**

Recopilar evidencia que demuestre la correlación entre la señal ECG medida con BiTalino y/o RespiBAN. Enfatizar el uso de la IA para predecir estados (estrés, baseline, meditación...), indicando precisión, sensibilidad y especificidad. Si se declara objetivo médico, se requeriría un estudio que refleje la eficacia del dispositivo en un número representativo de pacientes.

4. **Validación de software y control de versiones**

Documentar la **IA** (incluyendo arquitectura, entrenamiento, validación, dataset WESAD + RespiBAN, etc.). Asegurar la reproducibilidad y la conformidad con pautas de software médico. Describir la seguridad en la transmisión de datos entre BiTalino y la aplicación que ejecuta la IA (p. ej., cifrado de datos o integridad de la señal).

5. **Instrucciones de uso y etiquetado**

Incluir las advertencias pertinentes (limitaciones del uso en entornos electromagnéticos fuertes, personal sanitario o usuario final, etc.). Recomendaciones para la limpieza y el mantenimiento de BiTalino (electrodos) o accesorios similares.

11.4. Formularios para solicitar el marcado CE

Una vez completada la documentación técnica, se **rellenan** los formularios conforme al MDR, que contienen:

1. **Datos del fabricante** y del responsable legal.
2. **Descripción comercial** de BitaHealth, con especificación de componentes (hardware + software).
3. **Lista de normas armonizadas** aplicadas para demostrar la conformidad (IEC 60601-1, ISO 14971, etc.).

4. **Clase del dispositivo** y la **regla** de clasificación invocada.
5. **Declaración UE de Conformidad** firmada, donde se atestigua que el producto cumple los requisitos del MDR.

Si es Clase IIa, se enviaría esta documentación al **Organismo Notificado** (TÜV, BSI, Dekra, etc.), que revisaría la conformidad técnica. Tras la aprobación, se puede colocar el **Marcado CE** y comercializar BitaHealth como dispositivo médico en el EEE (Espacio Económico Europeo).

11.5 Conclusiones en cuanto a la solicitud del marcado CE

BitaHealth, al combinar la funcionalidad de BiTalino y los datos de RespiBAN, ambos desarrollados por PLUX, con un **algoritmo de IA** para evaluar el estado de estrés, se convierte en un sistema más complejo que un mero kit de investigación. Si se pretende un uso médico concreto, se requiere:

- **Identificar** los reglamentos aplicables (MDR 2017/745).
- **Alinear** la clasificación del dispositivo (posible Clase IIa).
- **Elaborar** un expediente técnico completo, detallando hardware, IA, validaciones y gestión de riesgos.
- **Solicitar** la conformidad a un Organismo Notificado.

En este contexto, parte de la documentación la facilitaría el fabricante (PLUX), que ya podría haber desarrollado ciertos ensayos de seguridad eléctrica y de compatibilidad electromagnética para BiTalino/RespiBAN. Sin embargo, la responsabilidad final de la conformidad recae en la entidad que comercializa BitaHealth como producto sanitario, debiendo demostrar la seguridad y eficacia de su integración hardware-software.

12. Conclusiones de la creación de BitaHealth

Tras todo el proceso, se pueden extraer varias **conclusiones**:

1. La fase de **preprocesado** y extracción de características HRV es determinante para la calidad final del sistema.
2. El uso de un **dataset público** (WESAD) que contenga múltiples sujetos en diferentes condiciones de estrés/relajación permite un **aprendizaje supervisado** sólido.
3. La **reducción de dimensionalidad** (PCA) y el **balanceo de clases** (SMOTE) pueden incrementar significativamente el rendimiento, tal como demuestra la diferencia entre Dataset 4 y los tres primeros.
4. El **modelo k-NN** demostró un comportamiento competitivo frente a otros modelos más sofisticados. Con un 83% de precisión, puede resultar idóneo para aplicaciones iniciales de detección de estrés en tiempo casi real.
5. **Flask** es una opción liviana y versátil para exponer el modelo mediante API, y **React** brinda una experiencia de usuario moderna y altamente personalizable para la presentación de resultados.

13. Trabajo futuro

Algunas direcciones para **extender** y **mejorar** el proyecto incluyen:

- **Aumento del conjunto de datos**: Incluir más sujetos, condiciones y señales (EDA, EMG, temperatura) para lograr mayor robustez en la clasificación.
- **Análisis en tiempo real**: Implementar un módulo que reciba la señal ECG de forma continua para detectar estrés de forma **online**.
- **Enriquecimiento del modelo**: Probar arquitecturas más complejas (p. ej., LSTM o CNN en secuencias de ECG), así como más medidas de HRV (non-linear, fractal).
- **Validación clínica**: Involucrar a profesionales de la salud para diseñar protocolos experimentales y validar la utilidad práctica del sistema en entornos reales de alta exigencia (p. ej., urgencias hospitalarias).
- **Despliegue en la nube**: Llevar la aplicación a servicios como AWS, Azure o GCP, a fin de habilitar un acceso global y facilitar el escalado del servicio.

14. Bibliografía y recursos

- WESAD Dataset Paper: Schmidt et al. (2018).
- NeuroKit2: Open-source Python package for physiological signal processing.
- BiTalino Official Website
- Scikit-Learn Documentation
- Imbalanced-learn: Handling of Imbalanced Datasets in Python.
- Flask Documentation
- React Official Documentation
- Malik M. et al. (1996). “Heart rate variability: Standards of measurement, physiological interpretation, and clinical use.” *Circulation*, 93, 1043-1065.

Anexo: Resumen del flujo de trabajo

1. **Recolección de datos** (WESAD + muestras propias en .txt).
2. **Lectura y segmentación** de la señal ECG, limpieza y extracción de características HRV.
3. **Generación de CSV** con columnas de HRV y etiquetas (Label).
4. **Creación de versiones del dataset** con distintas estrategias de preprocesado (imputación, PCA, SMOTE...).
5. **Entrenamiento** de modelos (k-NN, Decision Tree, RandomForest, etc.) y comparación de métricas.

6. **Selección de k-NN** sobre Dataset 4 como modelo final.
7. **Despliegue en Flask**: carga de modelo .pkl, preprocesadores y funciones de extracción de HRV con *NeuroKit2*.
8. **Desarrollo en React**: interfaz para subir el .txt, recibir resultados e imprimir informes (PDF).

De esta manera, se completa el ciclo de vida de un proyecto de ciencia de datos, desde la adquisición y limpieza de información bruta hasta la entrega de una solución funcional en un entorno productivo.