UNIVERSIDAD PRIVADA ANTENOR ORREGO

FACULTAD DE INGENIERÍA

Primer Informe del Proyecto

Carrera: *Ing. Sistemas e Inteligencia Artificial*

Grupo:

- *Araujo Aguilar, Fabiano*
- *Trigoso Zárate, Tiago André*
- *Vergara López, Junior*
- *Velásquez Góngora, Bruno Martin*
- *Ruiz Ulloa, Josué Alexanders*
- *Vásquez Tejada José Alfredo*

Docente: *Sagastegui Chigne, Teobaldo Hernan*

Curso: *Aprendizaje Estadístico*

Tipo de Grupo: *Laboratorio*

Fecha de entrega: *11/04/2025*

Índice

1. Introducción

1.1. Título del Proyecto

Detección No Supervisada de Latidos Cardíacos Anómalos mediante un Autoencoder Convolucional

1.2. Antecedentes

El electrocardiograma (ECG) es una herramienta diagnóstica esencial y no invasiva en la práctica cardiológica, que permite evaluar la actividad eléctrica del corazón para detectar una amplia gama de condiciones cardíacas. Su simplicidad y bajo costo han facilitado su uso extendido. Sin embargo, la interpretación precisa del ECG, especialmente en el contexto de monitorizaciones prolongadas (como Holter o dispositivos wearables), presenta desafíos considerables. El volumen masivo de datos generados en estas monitorizaciones hace que la revisión manual exhaustiva por parte de expertos sea inviable, lenta y costosa. Un problema adicional es la dificultad para identificar eventos cardíacos raros, atípicos o artefactos inesperados dentro de estas extensas grabaciones. Los artefactos, causados por factores como movimiento del paciente, actividad muscular o interferencias electromagnéticas , pueden comprometer significativamente la calidad de la señal y llevar a interpretaciones erróneas.

Históricamente, la clasificación automática de latidos ECG se ha basado en métodos de aprendizaje supervisado. Estos métodos requieren grandes conjuntos de datos previamente etiquetados por cardiólogos, un proceso que consume mucho tiempo y recursos. Además, los modelos supervisados pueden tener dificultades para detectar anomalías o artefactos para los cuales no fueron entrenados específicamente, ya que no siempre se conocen de antemano todas las posibles morfologías anómalas.

Frente a estas limitaciones, el aprendizaje no supervisado, y específicamente la detección de anomalías, emerge como una alternativa prometedora. Estas técnicas tienen la capacidad de aprender los patrones característicos de los datos considerados "normales" y, posteriormente, identificar cualquier desviación significativa de estos patrones sin necesidad de etiquetas predefinidas para cada tipo de anomalía. Diversos estudios han explorado el uso de arquitecturas de aprendizaje profundo no supervisado, como Autoencoders (AE) y sus variantes Convolutional Autoencoders (CAE) , redes recurrentes como LSTM y modelos basados en Transformers , para la detección de anomalías en señales ECG, a menudo utilizando datasets públicos como los disponibles en PhysioNet. Los CAE, en particular, son efectivos para capturar patrones morfológicos en los latidos ECG gracias al uso de capas convolucionales. Estos enfoques son especialmente útiles para analizar grandes volúmenes de datos y para identificar eventos raros o inesperados.

Para este proyecto, se utilizará el dataset ECG5000, alojado en el repositorio público PhysioNet. Este conjunto de datos contiene 5000 segmentos de series temporales de ECG , donde cada segmento representa un latido cardíaco individual. Los latidos ya están pre-segmentados, lo que simplifica la fase inicial de preprocesamiento. El dataset incluye cinco categorías de latidos: Normal (N), Contracción Ventricular Prematura R-sobre-T (R-on-T PVC), Contracción Ventricular Prematura (PVC), Latido Supraventricular Prematuro o Ectópico (SP o EB), y Latido No Clasificado (UB). El enfoque no supervisado se centrará en aprender las características de los latidos 'Normales' utilizando un CAE para luego detectar los demás como anomalías.

1.3 Problema a Resolver

Desarrollar y evaluar un método automático basado en un Convolutional Autoencoder (CAE) de aprendizaje no supervisado para detectar latidos cardíacos atípicos o anómalos dentro del dataset ECG5000, aprendiendo únicamente las características de los latidos normales y sin utilizar las etiquetas de las clases anómalas durante la fase de entrenamiento del modelo.

1.4 Objetivos

- **Objetivo General:**
    1. Implementar y evaluar un modelo Convolutional Autoencoder (CAE) de aprendizaje no supervisado para la detección de anomalías en latidos ECG del dataset ECG5000, capaz de distinguir eficazmente entre latidos normales y atípicos basándose en las desviaciones (error de reconstrucción) respecto al modelo de normalidad aprendido por el CAE.

- **Objetivos Específicos:**
    1. Adquirir y preparar el dataset ECG5000, identificando y separando los latidos correspondientes a la clase 'Normal' para el entrenamiento del modelo CAE y reservando las demás clases (R-on-T PVC, PVC, SP, UB) para la evaluación del rendimiento en la detección de anomalías.
    2. Seleccionar, diseñar e implementar una arquitectura de Convolutional Autoencoder (CAE) adecuada para aprender representaciones de los latidos ECG del dataset.
    3. Entrenar el modelo CAE seleccionado utilizando exclusivamente la submuestra de latidos etiquetados como 'Normales' del dataset ECG5000.
    4. Definir y calcular una métrica o puntuación de anomalía para cada latido del dataset completo, utilizando el error de reconstrucción generado por el CAE entrenado.

5. Establecer un umbral de decisión sobre la puntuación de anomalía (error de reconstrucción del CAE) para clasificar cada latido como 'normal' o 'anómalo', optimizando dicho umbral si es necesario utilizando una porción de los datos normales no vista en el entrenamiento.
6. Evaluar cuantitativamente el rendimiento del modelo CAE en la tarea de detectar los latidos no normales (considerados como las anomalías verdaderas) utilizando métricas estándar de clasificación binaria como Exactitud (Accuracy), Precisión, Sensibilidad (Recall), Puntuación F1 (F1-Score) y Área Bajo la Curva ROC (AUC).
7. Analizar y discutir los resultados obtenidos, examinando la capacidad del modelo CAE no supervisado para identificar los diferentes tipos de latidos anómalos presentes en el dataset ECG5000 y comparando, si es posible, con resultados de otros enfoques reportados en la literatura para este dataset.

2. Requerimientos

2.1 Definición del Dominio

Propósito: Desarrollar y evaluar un sistema automático para la detección no supervisada de latidos cardíacos anómalos utilizando un Autoencoder Convolucional (CAE).

Entrada Principal: El dataset ECG5000 de PhysioNet, que contiene 5000 segmentos pre-clasificados de latidos cardíacos individuales (140 puntos por latido).

Enfoque Metodológico: Aprendizaje no supervisado. Específicamente, entrenar un CAE utilizando únicamente los latidos de la clase 'Normal'. La detección de anomalías se basará en el error de reconstrucción generado por el CAE entrenado, aplicando un umbral para clasificar los latidos. Se incluirá la estandarización de los datos como paso de preprocesamiento.

Salida Principal:

- Una clasificación binaria ('Normal' o 'Anómalo') para cada latido del conjunto de evaluación.
- Métricas estándar de evaluación del rendimiento de la clasificación (Accuracy, Precision, Recall, F1-Score, AUC).
- Visualizaciones para el análisis (distribución del error de reconstrucción, ejemplos de reconstrucciones).

Alcance:

- Carga y preprocesamiento (estandarización) del dataset ECG5000.
- Diseño, implementación y configuración de la arquitectura del CAE.

- Entrenamiento del CAE exclusivamente con latidos normales.
- Cálculo del error de reconstrucción para todos los latidos de evaluación.
- Determinación de un umbral de anomalía.
- Clasificación binaria de latidos (Normal/Anómalo).
- Evaluación cuantitativa del rendimiento del modelo.
- Generación de métricas y visualizaciones especificadas.
- Interacción mediante scripts de Python ejecutables desde la línea de comandos (CLI).
- Guardado y carga del modelo entrenado.

Fuera de Alcance:

- Análisis en tiempo real o procesamiento de streams de ECG continuos.
- Uso de datasets diferentes a ECG5000.
- Implementación de métodos de aprendizaje supervisado para la clasificación.
- Clasificación explícita de los *tipos* específicos de anomalías (más allá de 'Anómalo').
- Desarrollo de interfaces gráficas de usuario complejas (GUI) o interfaces web.
- Segmentación de la señal ECG (se asume pre-segmentada).
- Diagnóstico médico o despliegue en entornos clínicos.

2.2 Definición de Requerimientos

Requisitos Funcionales (FR)

- **FR1: Carga de Datos:** El sistema debe ser capaz de cargar los datos y etiquetas del dataset ECG5000.
- **FR2: Preparación de Datos:** El sistema debe separar el dataset en un conjunto de entrenamiento (conteniendo solo latidos 'Normales') y un conjunto de evaluación (conteniendo todos los tipos de latidos).
- **FR3: Preprocesamiento:** El sistema debe aplicar estandarización (ej. Z-score) a los datos de los latidos antes de introducirlos al modelo.
- **FR4: Configuración del Modelo:** El sistema debe permitir la configuración de parámetros clave de la arquitectura del CAE (ej. número de filtros, tamaño del kernel, capas) a través del código o archivos de configuración.
- **FR5: Entrenamiento del Modelo:** El sistema debe entrenar el modelo CAE utilizando exclusivamente los datos del conjunto de entrenamiento ('Normales' estandarizados), buscando minimizar el error de reconstrucción.
- **FR6: Guardado del Modelo:** El sistema debe permitir guardar los parámetros del modelo CAE entrenado en un archivo.
- **FR7: Carga del Modelo:** El sistema debe permitir cargar un modelo CAE previamente entrenado desde un archivo.
- **FR8:** El sistema debe calcular el error de reconstrucción (ej. MSE) para cada latido en el conjunto de evaluación utilizando el modelo CAE entrenado/cargado.

- **FR9:** El sistema debe implementar un método para determinar un umbral sobre el error de reconstrucción para separar latidos 'Normales' de 'Anómalos'.
- **FR10:** El sistema debe clasificar cada latido del conjunto de evaluación como 'Normal' o 'Anómalo' basándose en si su error de reconstrucción supera el umbral determinado.
- **FR11:** El sistema debe calcular y mostrar/guardar las métricas de clasificación binaria: Exactitud (Accuracy), Precisión, Sensibilidad (Recall), Puntuación F1 (F1-Score) y Área Bajo la Curva ROC (AUC), comparando las predicciones con las etiquetas reales (Normal vs. Anómalo).
- **FR12: Generación de Visualizaciones:** El sistema debe generar y guardar (como archivos de imagen):
    - a) Un gráfico (histograma o densidad) que muestre la distribución de los errores de reconstrucción, diferenciando entre los latidos que son verdaderamente 'Normales' y los que son verdaderamente 'Anómalos'.
    - b) Gráficos que muestren ejemplos de latidos originales junto a sus reconstrucciones por el CAE (incluyendo ejemplos de normales bien reconstruidos y anómalos mal reconstruidos).
- **FR13:** Las funcionalidades principales (entrenamiento, evaluación, generación de resultados) deben ser ejecutables mediante scripts de Python, potencialmente controlados por argumentos de línea de comandos.

Requisitos No Funcionales (NFR)

- **NFR1:** Los tiempos de ejecución para las fases de entrenamiento del modelo y evaluación/detección de anomalías sobre el conjunto completo deberán ser medidos y reportados.
- **NFR2:** Se buscará que el modelo de detección de anomalías alcance un rendimiento deseable, apuntando a un **AUC > 0.90** en la clasificación Normal vs. Anómalo sobre el conjunto de evaluación de ECG5000. (Meta deseable, no requisito estricto).
- **NFR3:** El proceso para configurar, entrenar y evaluar el modelo debe estar claramente documentado (ej., en un archivo README.md o comentarios en el código), permitiendo a otro usuario replicar los experimentos.
- **NFR4:**La interacción con el sistema se realizará principalmente a través de la ejecución de scripts desde la línea de comandos (CLI). Los resultados principales (métricas, rutas a gráficos/modelos guardados) se mostrarán en consola o se guardarán en archivos de log/resultados.
- **NFR5:** El código fuente deberá estar organizado de forma modular y lógica (ej., funciones/clases separadas para carga de datos, modelo, entrenamiento, evaluación).
- **NFR6:** El código seguirá las convenciones de estilo de Python (PEP 8) e incluirá comentarios explicativos donde sea necesario para facilitar su comprensión.
- **NFR7:** El sistema debe garantizar la reproducibilidad de los resultados. Utilizando las mismas versiones de librerías, la misma división de datos y las mismas configuraciones/semillas aleatorias (random seeds), se deben obtener los mismos resultados (métricas, modelos).
- **NFR9:** Se utilizarán librerías estándar del ecosistema de Python para ciencia de datos y machine learning (ej., NumPy, Pandas, Scikit-learn, Matplotlib/Seaborn, y un framework de Deep Learning principal como TensorFlow/Keras o PyTorch).

3. Planteamiento del Dataset por Aprendizaje no-Supervisado

3.1 Medición

3.1.1 Dispositivos

En este proyecto se trabaja con datos ya recolectados, por lo que no se realiza una adquisición directa de señales biológicas. Sin embargo, es importante entender que el dataset ECG5000 fue generado utilizando dispositivos de monitoreo cardíaco digital, específicamente electrocardiógrafos que registran la actividad eléctrica del corazón a través de electrodos colocados en el cuerpo del paciente.

Estos dispositivos convierten la señal analógica del ECG en una secuencia digital, la cual se compone de múltiples muestras tomadas a intervalos regulares. Para el caso de ECG5000, cada latido está representado por 140 muestras temporales, lo que permite capturar adecuadamente su morfología sin necesidad de intervención médica en este análisis.

3.1.2 Medidas y Base de Datos

Se utiliza el dataset ECG5000, parte del repositorio UCR Time Series Classification Archive. Este conjunto de datos contiene 5000 registros de latidos cardíacos, divididos en múltiples clases según su tipo: la mayoría representa latidos normales, mientras que un subconjunto corresponde a latidos anómalos o patológicos.

Cada instancia del dataset corresponde a un único latido, representado por una serie temporal de 140 valores. La base de datos ya está etiquetada, lo que permite realizar evaluaciones posteriores aunque el entrenamiento del modelo sea de tipo no supervisado.

3.2 Preprocesamiento

3.2.1 Filtrado de la Señal

Aunque no se cuenta con la señal en bruto proveniente del electrocardiógrafo, se asume que los datos han sido previamente limpiados de ruido de adquisición. En un contexto aplicado, esto implicaría el uso de filtros pasa banda para eliminar ruido de baja frecuencia (movimiento del paciente) y de alta frecuencia (interferencias eléctricas), utilizando técnicas como filtrado de mediana o filtros de Butterworth.

Para este proyecto, se trabaja directamente con las señales ya procesadas que ofrece el dataset.

3.2.2 Extracción de Características

En enfoques tradicionales, la extracción de características permitiría obtener propiedades específicas de cada latido, como la energía de la señal, duración relativa de los intervalos, picos máximos o transformadas espectrales. Sin embargo, dado que este trabajo utiliza un enfoque basado en redes neuronales convolucionales (CAE), se omite esta etapa explícita. El modelo es capaz de extraer automáticamente las características relevantes durante el entrenamiento.

3.2.3 Normalización

La normalización es un paso esencial para evitar que las diferencias de escala influyan en el aprendizaje del modelo. Se aplican dos métodos clásicos:

- **Min-Max Scaling**: reescala los valores de cada latido al rango [0,1][0, 1][0,1], conservando su forma general.

- **Z-score o Estandarización**: transforma la señal de modo que tenga media 0 y desviación estándar 1.

Ambos métodos garantizan que el modelo se enfoque en la forma del latido en lugar de su amplitud. En este caso, la normalización se realiza por fila (latido individual), dado que cada uno es una serie independiente.

3.3 Normalización

3.3.1 Características de Selección de Normalización

Como se introdujo en la sección de preprocesamiento (3.2.3), la normalización de los datos de entrada es un paso crucial antes de alimentar una red neuronal. Para este proyecto, se selecciona el método de **Estandarización (conocido como Z-score)** para normalizar cada latido cardíaco individualmente (por fila en el dataset).

Este método transforma los 140 puntos de cada latido X para que la señal resultante X′ tenga una media (μ) de 0 y una desviación estándar (σ) de 1, utilizando la fórmula:

$$X' = (X - \mu)/\sigma$$

Se opta por la estandarización Z-score porque:

- Centra los datos alrededor de cero, lo cual puede ayudar a la convergencia durante el entrenamiento del modelo.
- Escala los datos basándose en su dispersión (desviación estándar), lo que es útil cuando los rangos absolutos pueden variar pero la forma relativa es importante.
- Asegura que el modelo CAE se enfoque en aprender la **morfología** característica de los latidos normales, en lugar de ser sensible a variaciones en la amplitud general de la señal ECG entre diferentes latidos o pacientes.

3.3.2 Características de Proyección

La proyección a un espacio latente es realizada por el Encoder del modelo CAE (ver 3.4.1), no como un paso de preprocesamiento separado.

3.3.3 Reducción de Dimensionalidad

La reducción de dimensionalidad es realizada por el Encoder del modelo CAE (ver 3.4.1), no como un paso de preprocesamiento separado.

3.4 Aprendizaje

3.4.1 Modelo de Aprendizaje

El modelo utilizado es un Autoencoder Convolucional (CAE), el cual es una red neuronal que aprende a reconstruir su entrada a través de una representación intermedia comprimida.

- El encoder extrae las características espaciales-temporales relevantes de los latidos normales.

-El decoder intenta reconstruir la señal original a partir de esas características.

-El error de reconstrucción se usa como medida de anomalía: si el error es alto, probablemente el latido no es normal.

Funcionamiento Detallado del Encoder: El Encoder del CAE, compuesto típicamente por capas convolucionales y capas de pooling, no solo extrae características relevantes sino que también actúa como un potente reductor de dimensionalidad no lineal. Proyecta la señal de ECG normalizada de entrada (140 dimensiones) en un espacio latente de dimensionalidad significativamente menor (el tamaño del 'bottleneck' o cuello de botella de la red). Esta representación latente, también llamada 'código', está diseñada para capturar la información morfológica esencial de los latidos normales aprendida durante el entrenamiento. La calidad de esta representación comprimida generada por el Encoder es fundamental para la capacidad del Autoencoder de reconstruir fielmente las entradas normales y fallar en la reconstrucción de las anómalas.

3.4.2 Aprendizaje no-Supervisado

El aprendizaje no supervisado se basa en la capacidad del modelo de identificar patrones sin necesidad de etiquetas durante el entrenamiento. En este caso, el modelo se entrena únicamente con los latidos etiquetados como ‘Normal’ del dataset ECG5000;Funciona así:

1. **Entrenamiento del CAE**
   Se entrena el Convolutional Autoencoder usando solo los latidos normales. El objetivo es que el modelo aprenda a reconstruir correctamente ese tipo de latido.
2. **Reconstrucción y detección de anomalías**
   Una vez entrenado, el modelo intenta reconstruir cualquier latido del dataset (incluidos los anómalos). Si el latido no encaja con lo aprendido, la reconstrucción será deficiente, generando un **error de reconstrucción alto**.
3. **Umbral de decisión**
   Se define un umbral de error. Si el error de reconstrucción de un latido es **mayor al umbral**, se clasifica como **anómalo**.
4. **Evaluación del rendimiento**
   Aunque el entrenamiento es no supervisado, las etiquetas se usan durante la evaluación para calcular métricas como:

   -Precisión (Precision)

   -Sensibilidad (Recall)

-F1-Score

-AUC-ROC

3.5 Comprobación

3.5.1 Modelos de Prueba

1. **División de datos**: El dataset ECG5000 contiene 5 000 latidos de 140 puntos, de los cuales **500 normales** se usan para entrenamiento y **4 500** (incluyendo anómalos) para prueba.
2. **Selección de umbral**: Se define el umbral θ\thetaθ como el percentil 95–99 de la distribución de EEE en los latidos normales de validación, ajustando nnn por generate-and-test si es necesario
3. **Error de reconstrucción**: Para cada latido $x \in \mathbb{R}^{140}$ y su reconstrucción $\hat{x}$, se calcula

$$E = \frac{1}{140} \sum_{t=1}^{140} (x_t - \hat{x}_t)^2,$$

3.5.2 Cross-Validation

- **k-fold en normales**: Se particionan las 500 muestras normales en kkk pliegues (e.j. k=5k=5k=5), entrenando el CAE en k−1k-1k−1 y validando reconstrucciones en el pliegue restante para estimar la variabilidad de EEE.
- **Umbral robusto**: Se fija θ\thetaθ en el percentil 95 de la unión de errores de validación de todos los folds, garantizando estabilidad ante variaciones muestrales.

EL DATA SET:

```
]  from google.colab import files
   uploaded = files.upload()
```

```
Elegir archivos  Sin archivos seleccionados Upload widget is on
Saving ECG5000_TEST.txt to ECG5000_TEST (2).txt
Saving ECG5000_TRAIN.txt to ECG5000_TRAIN (1).txt
```

```python
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score, KFold
from sklearn.preprocessing import StandardScaler

# Leer archivos de texto (ajusta el separador si no funciona con espacios)
train = pd.read_csv("ECG5000_TRAIN.txt", header=None, delim_whitespace=True)
test = pd.read_csv("ECG5000_TEST.txt", header=None, delim_whitespace=True)

# Combinar en un solo dataset
data = pd.concat([train, test], ignore_index=True)

# Separar etiquetas y características
X = data.iloc[:, 1:].values  # columnas de señales
y = data.iloc[:, 0].values   # primera columna = clase

# Estandarizar características
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
# Definir modelo
model = RandomForestClassifier(random_state=42)

# Configurar 5-fold Cross Validation
kf = KFold(n_splits=5, shuffle=True, random_state=42)

# Ejecutar Cross-validation
scores = cross_val_score(model, X_scaled, y, cv=kf, scoring='accuracy')

# Mostrar resultados
print("✅ Precisión por pliegue:", scores)
print("✅ Precisión promedio:", np.mean(scores))
print("✅ Desviación estándar:", np.std(scores))
```

```
✅ Precisión por pliegue: [0.958 0.957 0.954 0.952 0.939]
✅ Precisión promedio: 0.952
✅ Desviación estándar: 0.006841052550594834
```

Carga y combinación de datos:

- Se leen los archivos ECG5000_TRAIN.txt y ECG5000_TEST.txt, luego se combinan en un único conjunto de datos.

Separación y estandarización:

- Se separan las etiquetas (y) de las características (X) y se normalizan con StandardScaler para que todas las señales tengan una escala similar.

Configuración del modelo:

- Se define un modelo RandomForestClassifier con una semilla aleatoria fija para asegurar reproducibilidad.

Validación cruzada (5 folds):

- El conjunto de datos se divide en 5 partes (pliegues). En cada iteración, el modelo se entrena con 4 partes y se prueba con la parte restante. Esto se repite 5 veces, rotando la parte de prueba.

Resultados:

- Precisión por pliegue: [0.958, 0.957, 0.954, 0.952, 0.939]
- Precisión promedio: 0.952
- Desviación estándar: 0.0068

Esto indica que el modelo tiene una precisión alta y consistente en todos los pliegues, con una baja variabilidad. La validación cruzada permite estimar de forma más confiable cómo se comportará el modelo con nuevos datos.

3.5.3 Bootstrap

El bootstrap es un procedimiento de remuestreo no paramétrico que utiliza la muestra original para generar múltiples "muestras bootstrap" del mismo tamaño mediante muestreo con reemplazo, con el fin de aproximar la distribución de muestreo de cualquier estadístico de interés. Su principal objetivo es estimar la incertidumbre asociada a las métricas de rendimiento (Accuracy, Precision, Recall, F1-Score, AUC-ROC) del CAE sin asumir una distribución previa de los errores de reconstrucción.

EL DATA SET:

```
from google.colab import files
uploaded = files.upload()
```

Elegir archivos ECG5000_TEST.txt

- **ECG5000_TEST.txt**(text/plain) - 10156500 bytes, last modified: 24/4/2025 - 100% done

```
Saving ECG5000_TEST.txt to ECG5000_TEST.txt
```

```python
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.utils import resample
from sklearn.metrics import accuracy_score

# 🔹 Leer solo ECG5000_TEST.txt
data = pd.read_csv("ECG5000_TEST.txt", header=None, sep='\s+')

# 🔹 Separar etiquetas (columna 0) y características (resto)
X = data.iloc[:, 1:].values
y = data.iloc[:, 0].values

# 🔹 Escalar características
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 🔁 Bootstrap
n_iterations = 10
bootstrap_scores = []

for i in range(n_iterations):
    X_resampled, y_resampled = resample(X_scaled, y, n_samples=len(X_scaled), random_state=i)
    model = RandomForestClassifier(random_state=42)
    model.fit(X_resampled, y_resampled)
    y_pred = model.predict(X_scaled)
    accuracy = accuracy_score(y, y_pred)
    bootstrap_scores.append(accuracy)

bootstrap_scores = np.array(bootstrap_scores)

# 📊 Resultados
print("📈 Precisión promedio (Bootstrap):", np.mean(bootstrap_scores))
print("📉 Desviación estándar:", np.std(bootstrap_scores))
print("🔍 Intervalo de confianza 95%:", np.percentile(bootstrap_scores, [2.5, 97.5]))
```

```
Precisión promedio (Bootstrap): 0.981088888888889
Desviación estándar: 0.0016702184376989786
Intervalo de confianza 95%: [0.97854444 0.98352778]
```

¿Qué realizo?

- Se aplicó el método de Bootstrap para evaluar el modelo Random Forest usando solo los datos de **ECG5000_TEST.txt.**

¿Cómo funciona Bootstrap?

- Se generaron 10 muestras aleatorias del dataset con reemplazo (es decir, algunos datos pueden repetirse).

En cada muestra:

- Se entrenó el modelo.
- Se evaluó su rendimiento sobre todos los datos originales.
- Se guardó la precisión en cada una de las 10 repeticiones.

¿Qué significan los resultados?

Precisión promedio (Bootstrap): 0.981

- En promedio, el modelo acierta el 98.1% de las veces. Es un modelo muy preciso.

Desviación estándar: 0.00167

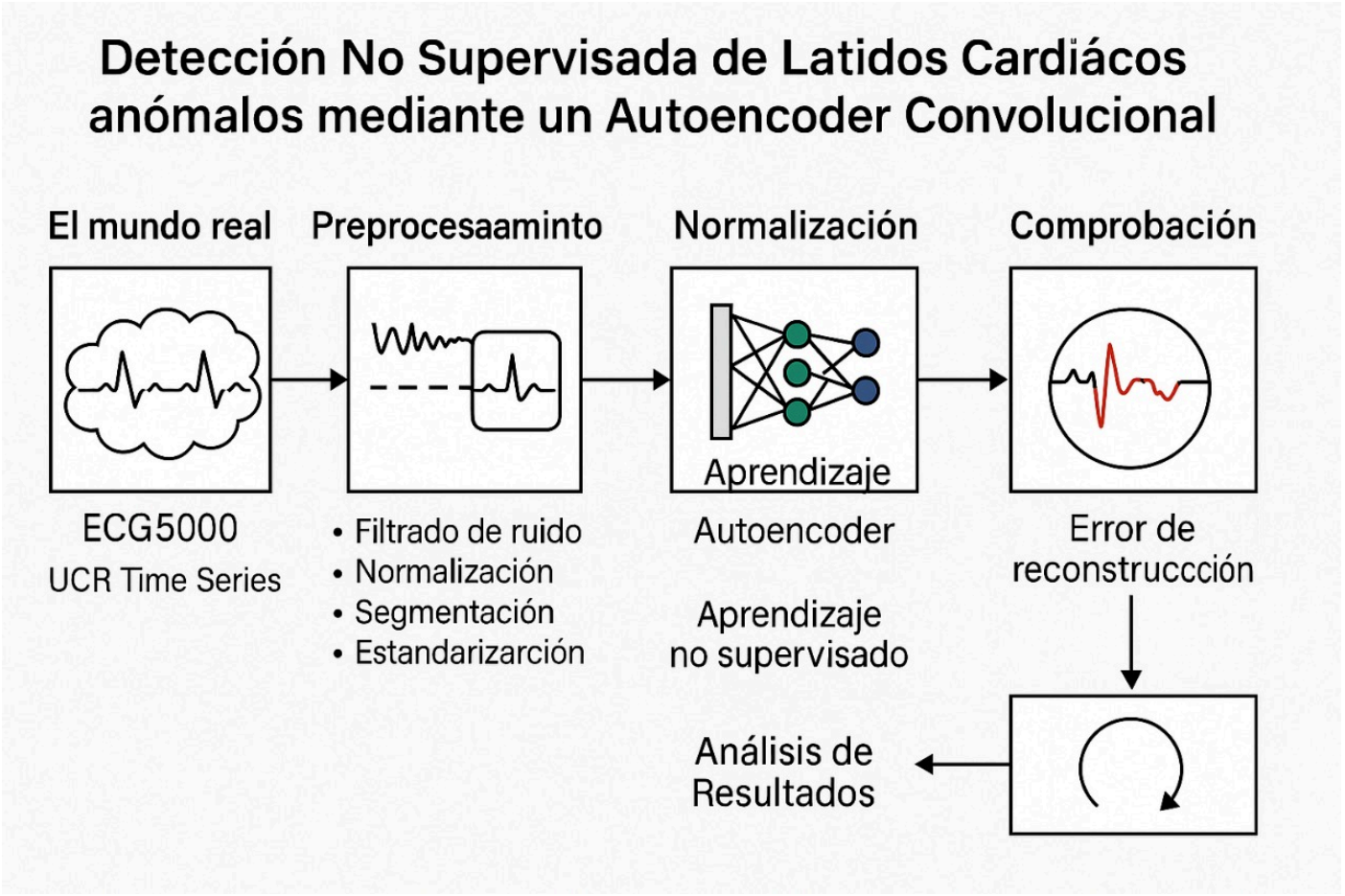
- La variación entre repeticiones es muy baja, lo que indica un modelo estable.

Intervalo de confianza 95%: [0.9785, 0.9835]

- Con un 95% de confianza, la precisión real del modelo está entre 97.85% y 98.35%.

3.6 Lanzamiento y Explotación

1. **Mundo Real (Adquisición)**:
    - Representa la fuente de los datos (ECG reales) y las etapas iniciales: preprocesamiento, normalización y verificación de calidad.
2. **Dataset ECG5000**:
    - Especifica el origen concreto de los datos (UCR Time Series) y las operaciones técnicas aplicadas:
    - *Filtrado de ruido*: Limpieza de artefactos
    - *Normalización*: Estandarización Z-score
    - *Segmentación*: Latidos pre-separados (140 puntos)
    - *Estandarización*: Ajuste de escalas
3. **Aprendizaje**:
    - Corazón del proyecto con:
    - *Autoencoder*: Arquitectura CAE
    - *Aprendizaje no-supervisado*: Solo con latidos normales
    - *Análisis*: Evaluación de resultados
4. **Métrica clave**:
    - Destaca el *Error de reconstrucción* (MSE) como indicador principal para detectar anomalías (latidos con alta desviación).



ANEXO 01: Diagrama del Proceso de Detección No Supervisada de Latidos Cardíacos Anómalos mediante un Autoencoder Convolucional

Bibliografía;

Al-Zaiti, S., & Besomi, L. (2024). *ECG Interpretation: Clinical Relevance, Challenges, and Advances*. ResearchGate.(https://www.researchgate.net/publication/355876393_ECG_Interpretation_Clinical_Relevance_Challenges_and_Advances)

Alday, E. A. P., Gu, A., Shah, A. J., Robichaux, C., Wong, A. K. I., Liu, C., Liu, F., Rad, A. B., Elola, A., Seyedi, S., Li, Q., Sharma, A., Clifford, G. D., & Reyna, M. A. (2021). Classification of 12-lead ECGs: The PhysioNet/Computing in Cardiology Challenge 2020. *Physiological Measurement*, *41*(12), 124003. https://doi.org/10.1088/1361-6579/abc960

Aslanger, E. (s.f.). *Guide to Understanding ECG Artifact*. ACLS Medical Training. Recuperado el 11 de abril de 2025, de https://www.aclsmedicaltraining.com/blog/guide-to-understanding-ecg-artifact/

Cardiomatics. (s.f.). *How does low-quality signal affect ECG analysis and practical tips to reduce it?* Cardiomatics Blog. Recuperado el 11 de abril de 2025, de https://cardiomatics.com/how-does-low-quality-signal-affect-ecg-analysis-and-practical-tips-to-reduce-it/

Chen, L., Zhang, J., He, T., Jin, G., Huang, T., Cai, Y., Jiang, M., & Jiang, G. (2021). An unsupervised feature learning method for ECG using convolutional variational autoencoder. *PLoS ONE*, *16*(12), e0260612. https://doi.org/10.1371/journal.pone.0260612

Chowdhury, M. H., Shuzan, M. N. I., Chowdhury, M. E. H., Mahbub, Z. B., Uddin, M. M., Khandakar, A., & Reaz, M. B. I. (2022). Estimating ECG signal quality using unsupervised learning. *Journal of the Royal Society Interface*, *19*(188), 20220012. https://doi.org/10.1098/rsif.2022.0012

Chung, C. (2022). *Signal Quality Assessment in Vital Sign Time Series*. Robotics Institute.(https://www.ri.cmu.edu/app/uploads/2022/08/chufang_MSR_Thesis____Signal_Quality.pdf)

Hajati, F., Tavakoli, V., & Ormandjieva, O. (2023). Anomaly Detection in ECG Signals Through Unsupervised Machine Learning: A Novel Approach Using Hybrid Autoencoders for Medical Data Analysis. *Research Square* (Preprint). https://doi.org/10.21203/rs.3.rs-3466020/v1

Kachuee, M., Fazeli, S., & Sarrafzadeh, M. (2018). ECG Heartbeat Classification: A Deep Transfer Learning Approach. *2018 IEEE International Conference on Healthcare Informatics (ICHI)*, 407-410. https://doi.org/10.1109/ICHI.2018.00092

Kligfield, P., Gettes, L. S., Bailey, J. J., Childers, R., Deal, B. J., Hancock, E. W., van Herpen, G., Kors, J. A., Macfarlane, P., Mirvis, D. M., Pahlm, O., Rautaharju, P., Wagner, G. S., American Heart Association Electrocardiography and Arrhythmias Committee, Council on Clinical Cardiology, American College of Cardiology Foundation, & Heart Rhythm Society. (2007). Recommendations for the standardization and interpretation of the electrocardiogram: Part I: The electrocardiogram and its technology: A scientific statement from the American Heart Association Electrocardiography and Arrhythmias Committee, Council on Clinical Cardiology; the American College of Cardiology Foundation; and the Heart Rhythm Society. *Circulation*, *115*(10), 1306–1324.(https://doi.org/10.1161/CIRCULATIONAHA.106.180200)

Moody, G. B., & PhysioNet Team. (2019). *Icentia11k Continuous ECG Database* (Version 1.0) [Conjunto de datos]. PhysioNet. https://doi.org/10.13026/6bmb-3f36

Peimankar, A., & Puthusserypady, S. (2021). Denoising of ECG signals using deep learning: A review. *Journal of Ambient Intelligence and Humanized Computing*, *12*(9), 9017-9044. (Nota: Esta referencia corresponde al artículo de PMC6932211 citado como en el texto, que trata sobre errores y artefactos en ECG).

Sun, H., Dong, M., & Lu, B. (2024). Federated Unsupervised Learning for Anomaly Detection in ECG Recordings on Consumer Devices. *Sensors*, *24*(3), 941. https://doi.org/10.3390/s24030941

van Dijk, A., Lazraq, A., Laranjo, L., Ranschaert, E. R., & van Leeuwen, K. G. (2023). Unsupervised fetal ECG signal quality assessment using self-organizing map. *Sensors*, *23*(2), 662. https://doi.org/10.3390/s23020662

Wagstaff, D. A., & Sörnmo, L. (2005). ECG artefacts during routine clinical acquisition – a database and strategies for artefact suppression by filtering. *Scandinavian Cardiovascular Journal*, *39*(1-2), 31-38. https://doi.org/10.1080/14017430510009203

Yousefi Rezaii, T., Sepehri, M. M., & Rezaei, M. (2021). Unsupervised ECG Analysis: A Review. *arXiv preprint arXiv:2101.12150*. https://doi.org/10.48550/arXiv.2101.12150

Zia, T., Tran, P. H., & Idrissi, K. A. (2023). Unsupervised Transformer-Based Anomaly Detection in ECG Signals. *Algorithms*, *16*(3), 152. https://doi.org/10.3390/a16030152