**FACULTAD DE INGENIERÍA**

**BIOINGENIERÍA**

Bioseñales y Sistemas

Valentina Garcia Obando C.C.1000539432
Juan José Alzate Molina. C.C.1007232151

# Informe Proyecto 4: Clasificación de señales ECG

1. **Revisión teórica. Del artículo: https://www.nature.com/articles/s41598-020-59821-7 Hacer un resumen de la sección extracción de características (Features extraction) y discutir desde el artículo u otras referencias como se hace cuando desaparecen ciertas formas de ondas en el complejo PQRS debido a alguna enfermedad**

El artículo **"Optimal Multi-Stage Arrhythmia Classification Approach"** describe un método avanzado para la clasificación de arritmias basado en ECGs multicanal. En la sección de extracción de características, se presentan las siguientes ideas clave:

- **Desafíos y características básicas**: Las características de los ECG, como la morfología de las ondas, varían significativamente entre individuos y dentro del mismo paciente. Problemas comunes incluyen la distorsión de ondas por ruido o condiciones cardíacas adicionales, y la variabilidad debida a factores como género, edad y raza.
- **Método propuesto**: Utiliza medidas como la altura, la prominencia y el ancho de las ondas P, Q, R, S y T, además de las relaciones entre picos y valles (e.g., proporciones entre diferencias de altura y tiempo). Los datos se normalizan para unificar las escalas de amplitud y mejorar la consistencia entre muestras. Introduce una tabla de frecuencia empírica que captura distribuciones de las características mencionadas, permitiendo un enfoque robusto y transparente para la extracción.
- **Ventajas del enfoque**: Su diseño explora 11 combinaciones distintas de características, desde medidas básicas hasta 39,830 variables para escenarios complejos. Este método supera limitaciones de otros enfoques como las transformadas de Fourier, que pierden información temporal, y las redes neuronales, que producen características menos interpretables.

Discusión sobre ondas desaparecidas en el complejo PQRS debido a enfermedades

La desaparición o distorsión de ondas en el complejo PQRS está relacionada con diversas condiciones cardíacas. Por ejemplo:

- **Fibrilación auricular (AFIB)**: Las ondas P desaparecen y son reemplazadas por ondas de fibrilación de baja amplitud, sin una relación clara con el ritmo QRS. Este fenómeno dificulta la detección de patrones regulares, exigiendo análisis más avanzados de relaciones entre picos y segmentos.
- **Bloqueos de rama (LBBB, RBBB)**: Las ondas QRS se ensanchan y deforman significativamente, lo que altera las proporciones típicas y puede confundir algoritmos de clasificación que no consideren esta distorsión.
- **Isquemia miocárdica**: Las ondas T pueden aplanarse o invertirse, mientras que el segmento ST se eleva o deprime, afectando los indicadores temporales y de amplitud usados en modelos predictivos.

El artículo analiza cómo su enfoque permite manejar estas distorsiones utilizando un enfoque de extracción de características robustas. Sin embargo, también se podrían considerar estudios adicionales, como el uso de redes neuronales para interpretar patrones de variabilidad en señales distorsionadas

2. **Herramientas computacionales. Hacer un minitutorial del uso de la herramienta NeuroKit (https://neuropsychology.github.io/NeuroKit/index.html) orientado al análisis de señales ECG**

**NeuroKit** es una librería de Python diseñada para facilitar el procesamiento y análisis de señales fisiológicas, incluyendo el ECG. Aquí tienes un tutorial paso a paso para orientarte en su uso en el análisis de señales ECG.

1. Instalación: Asegúrate de instalar la librería en tu entorno de Python. Usa el siguiente comando:

```
pip install neurokit2
```

2. Importar las librerías necesarias: Importa las librerías básicas para trabajar con datos y NeuroKit:

```
import neurokit2 as nk
import matplotlib.pyplot as plt
```

3. Cargar y visualizar una señal ECG

```
# Generar una señal ECG sintética
ecg_signal = nk.ecg_simulate(duration=10, sampling_rate=500)  # 10 segundos, 500 Hz

# Visualizar la señal ECG
plt.plot(ecg_signal)
plt.title("ECG Simulada")
plt.xlabel("Tiempo (muestras)")
plt.ylabel("Amplitud")
plt.show()
```

4. Procesamiento de la señal ECG

```
# Procesar la señal ECG
ecg_cleaned = nk.ecg_clean(ecg_signal, sampling_rate=500)

# Visualizar la señal original y limpia
plt.figure(figsize=(10, 5))
plt.plot(ecg_signal, label="ECG Original")
plt.plot(ecg_cleaned, label="ECG Limpia")
plt.legend()
plt.show()
```

5. Segmentación y extracción de características

```
# Detectar picos R
ecg_peaks, info = nk.ecg_peaks(ecg_cleaned, sampling_rate=500)

# Visualizar los picos detectados
nk.events_plot(info["ECG_R_Peaks"], ecg_cleaned)
plt.title("Picos R detectados")
plt.show()

# Extraer características de la señal
ecg_features = nk.ecg_intervalrelated(ecg_peaks, sampling_rate=500)
print(ecg_features.head())  # Mostrar las primeras filas
```

6. Análisis avanzado

- **Detección de ondas P, QRS, T**:

```
# Delimitación de ondas
signals, waves = nk.ecg_delineate(ecg_cleaned, ecg_peaks, sampling_rate=500, method="dwt")

# Visualizar las ondas detectadas
plt.plot(ecg_cleaned, label="ECG Limpia")
plt.scatter(waves["ECG_R_Peaks"], ecg_cleaned[waves["ECG_R_Peaks"]], label="Picos R", color="red")
plt.legend()
plt.show()
```

- **Frecuencia cardíaca promedio**:

```
# Frecuencia cardíaca
hr = nk.ecg_rate(info["ECG_R_Peaks"], sampling_rate=500, desired_length=len(ecg_cleaned))
plt.plot(hr)
plt.title("Frecuencia Cardíaca Instantánea")
plt.xlabel("Tiempo (muestras)")
plt.ylabel("Frecuencia cardíaca (bpm)")
plt.show()
```

7. Exportar los resultados

```
# Exportar a un archivo CSV
ecg_features.to_csv("ecg_features.csv", index=False)
print("Características exportadas a ecg_features.csv").
```

Recursos adicionales

- **Documentación oficial:** NeuroKit2 Docs
- **GitHub de NeuroKit:** Repositorio

3. **Con las características que extrae el neurokit y las discutidas en el punto 1 generar un conjunto de características para el presente trabajo. Incluir la característica de frecuencia de potencia máxima del Proyecto 3 (10%). No incluir más de 15 características ni menos de 5, justificar la selección (5%).**

Se utiliza para la clasificación posterior la métrica de fMP (Frecuencia de máxima potencia) y del Neurokit se toman 9 métricas principales basados en las características que puedan facilitar la diferenciación de ambas patologías y también la presencia de valores nulos o atípicos para el conjunto de datos utilizado. Las métricas adicionales son las siguientes:

- **ECG_Rate_Mean:** Mide la frecuencia cardíaca promedio. Es útil porque la AFIB tiende a tener una frecuencia cardíaca más alta y variable que la SB.

- **HRV_SDNN:** Estándar de desviación de todos los intervalos NN (latidos normales). Refleja la variabilidad total y ayuda a distinguir entre ritmos regulares (SB) e irregulares (AFIB).

- **HRV_RMSSD**: Raíz cuadrada de la media de las diferencias al cuadrado entre intervalos sucesivos NN. Captura variaciones rápidas en la frecuencia cardíaca, más prominentes en AFIB.

- **HRV_pNN50**: Porcentaje de diferencias consecutivas de intervalos NN que superan los 50 ms. Suele ser menor en AFIB debido a la variabilidad caótica.

- **HRV_SD1:** Desviación estándar en el eje corto del diagrama de Poincaré. Indica variabilidad a corto plazo y es útil para detectar la irregularidad de AFIB.

- **HRV_SD2:** Desviación estándar en el eje largo del diagrama de Poincaré. Mide tendencias a largo plazo y puede ser mayor en AFIB debido a la desorganización.
- **HRV_SD1SD2:** Relación entre SD1 y SD2. Es un buen indicador de irregularidad: valores más bajos sugieren un ritmo más caótico, como en AFIB.

- **HRV_CVNN:** Coeficiente de variación de los intervalos NN. Muestra la variabilidad relativa; valores altos son característicos de AFIB.

- **HRV_HTI:** Índice triangular de HRV. Suele ser menor en AFIB porque refleja una distribución más desordenada de los intervalos.

Cabe resaltar que el método utilizado fue ecg_intervalrelated(). Sin embargo, gran parte de los datos de SB (más del 90%) presentaron incompatibilidad con el método pues el tiempo de muestreo de la señal era demasiado corto para la evaluación de métricas de alta complejidad, y la posible solución (además de modificar el código fuente de la librería) sería tomar nuevas señales con la cantidad de datos necesarios a esa frecuencia de muestreo.

4. **Código y análisis de resultados, donde se discutan por los menos tres diferentes arquitecturas de red (10%) y las matrices de confusión obtenidas (10%), de una red neuronal que permita la clasificación de las dos patologías cardiacas.**

Para el tratamiento previo al entrenamiento de los modelos se balanceó la cantidad de sujetos de una categoría y la otra (101 por patología) con el fin de asegurar la presencia de suficientes datos de ambas patologías en el conjunto de datos de entrenamiento. Se realizó el entrenamiento, fit y test con 3 redes neuronales distintas, todas generadas con el modelo MLPClassifier

- **Red Sencilla (1 capa oculta, 16 neuronas)**

Considerando la cantidad de métricas (10), la cual es igual a la cantidad de neuronas en la primera capa, y considerando que el dataset utilizado es de tamaño reducido, se toma la desición de empezar con una red de baja complejidad que reduzca el riesgo de sobreentrenamiento con el conjunto de datos, las matrices de confusión obtenidas fueron las siguientes:

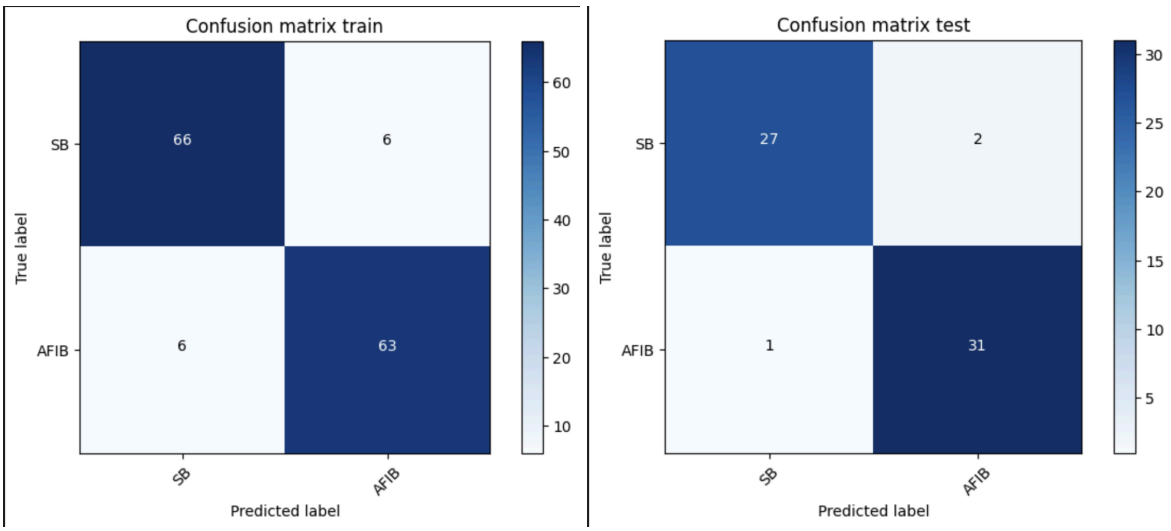


Figura 1. Matrices de confusión para la primera arquitectura (1 Capa oculta, 16 Neuronas)

Se puede apreciar un resultado muy positivo con accuracy del 91% para el entrenamiento y 95% para el test. La presencia de pocas neuronas permite dar cuenta de una selección adecuada de métricas, que permiten al algoritmo entender la relaciones entre los 10 parámetros presentados y predecir con alta fiabilidad.

- **Red Intermedia (2 capas ocultas, 32 y 16 neuronas)**

Esta segunda arquitectura permite aumentar la capacidad del algoritmo de encontrar relaciones no lineales más complejas entre las métricas sin aumentar en un alto grado la complejidad y carga computacional. Las dos capas permiten además modelar relaciones jerárquicas sin exceder la capacidad del modelo.

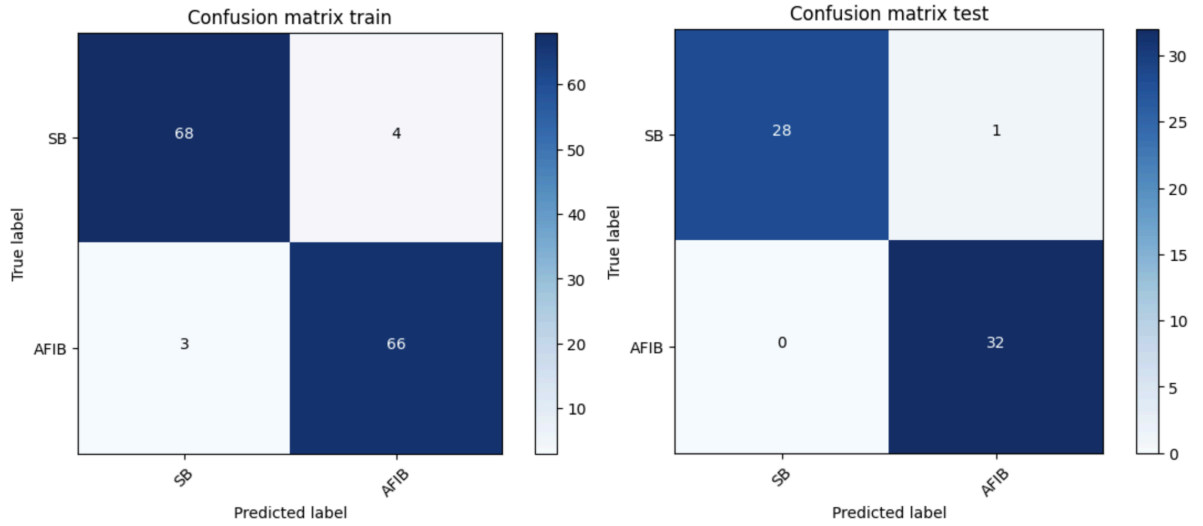


Figura 2. Matrices de confusión para la segunda arquitectura (2 capas oculta, 32 y 16 Neuronas)

Aumenta el grado de precisión para ambos grupos y en ambos subconjuntos de datos, alcanzando un 95% para entrenamiento y 98% para test, podemos todavía suponer que no se debe a un sobreentrenamiento, por la excelente capacidad predictiva, sin embargo sería interesante contrastar resultados para este modelo con un conjunto más grande de datos.

- **Red Avanzada (3 capas ocultas, 64, 32 y 16 neuronas)**

Para esta última red se aumenta exponencialmente la complejidad y se llega cerca del límite recomendado para este modelo de ML. Esta arquitectura permite responder a datos de mayor complejidad o relaciones no lineales. La principal ventaja de este modelo es que es escalable a datasets de mayor complejidad, o de mayor tamaño que el estudiado en este caso específico.

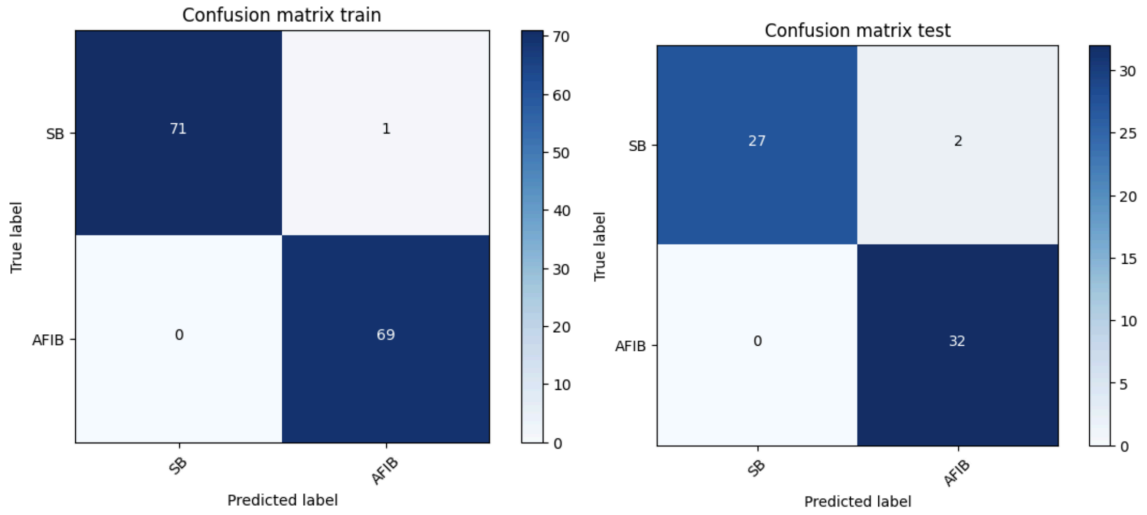


Figura 3. Matrices de confusión para la tercera arquitectura (3 capas oculta, 64, 32 y 16 Neuronas)

Para esta tercera arquitectura podemos apreciar un mayor accuracy para el conjunto de entrenamiento que para el test (lo cual no es señal de alarma), sin embargo, un accuracy del 99% para el subconjunto de entrenamiento es un indicio de posible sobreentrenamiento del modelo, esto tendría sentido pues la complejidad ya es muy alta para un dataframe de reducidas proporciones como el construido para este caso.

5. **Consultar cómo funciona, realizar y discutir un ejemplo con los datos, del algoritmo de K means y máquinas de soporte vectorial (SVM)**

**Algoritmo K-means**

**¿Qué es K-means?** K-means es un algoritmo de **aprendizaje no supervisado** que agrupa datos en un número k de clusters (grupos) con base en sus características. Este algoritmo funciona iterativamente para minimizar la variabilidad intra-cluster y maximizar la variabilidad inter-cluster [1], [2].

Pasos básicos del algoritmo:

- Inicializar k centroides (puntos de referencia iniciales).
- Asignar cada punto de datos al clúster cuyo centroide esté más cercano.
- Actualizar los centroides como el promedio de todos los puntos asignados al clúster.
- Repetir los pasos 2 y 3 hasta que los centroides ya no cambien significativamente o se alcance el máximo de iteraciones.

**Ejemplo con datos de ECG:** Para implementar K-means en Python, se utiliza la librería Scikit-learn, que incluye funciones predefinidas para su aplicación [4]. En este caso, se agrupan características del ECG como la frecuencia cardíaca (HR), el intervalo PR y la duración del complejo QRS. La elección del número óptimo de clusters (k) se puede realizar utilizando el método del codo, que es ampliamente discutido en la literatura [5].

```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import pandas as pd
# Supongamos que tenemos un DataFrame con características de ECG
data = pd.read_csv("ecg_features.csv")  # Contiene columnas como HR, PR_interval, QRS_duration, etc.
# Seleccionar las características más relevantes
X = data[['HR', 'PR_interval', 'QRS_duration']]  # Frecuencia cardíaca, intervalo PR y duración QRS
# Aplicar K-means
kmeans = KMeans(n_clusters=2, random_state=42)  # Dos clusters: SB y AFIB
kmeans.fit(X)
# Agregar etiquetas al DataFrame
data['Cluster'] = kmeans.labels_
# Visualizar resultados
plt.scatter(X['HR'], X['QRS_duration'], c=data['Cluster'], cmap='viridis', label="Clusters")
plt.xlabel("Frecuencia Cardíaca (HR)")
plt.ylabel("Duración QRS")
plt.title("Clustering K-means en ECG")
plt.show()
```

**Discusión de resultados:** K-means es útil para identificar patrones en datos no etiquetados, como las características del ECG. Sin embargo, es sensible a la inicialización de centroides y requiere una elección cuidadosa de k. Estas limitaciones son señaladas por Liu et al., quienes comparan K-means con otros

algoritmos de clustering en términos de robustez [3]. Además, el método del codo es una herramienta práctica para seleccionar k, evaluando la variabilidad intra-cluster [5].

**Máquinas de Soporte Vectorial (SVM)**

**¿Qué es SVM?** SVM es un algoritmo de **aprendizaje supervisado** que clasifica datos al encontrar un hiperplano que separa las clases con el mayor margen posible. Este enfoque es particularmente efectivo en problemas linealmente separables y se puede extender a datos no lineales mediante el uso de kernels [6], [7]. Los SVM han sido utilizados con éxito en aplicaciones biomédicas, incluyendo el análisis de señales ECG [8]..

Pasos básicos del algoritmo:

- Identificar un hiperplano que divida las clases de datos.
- Maximizar el margen entre el hiperplano y los puntos de datos más cercanos (vectores de soporte).
- Si los datos no son linealmente separables, transforma los datos a un espacio de mayor dimensión con un kernel.

**Ejemplo con datos de ECG:** En este caso, se utiliza un SVM con kernel RBF (Radial Basis Function) para clasificar datos etiquetados del ECG, como bradicardia sinusal (SB) y fibrilación auricular (AFIB). Scikit-learn proporciona una implementación eficiente de SVM que permite ajustar parámetros como C y gamma para optimizar el modelo [9]. Además, Patel discute cómo la elección de los hiperparámetros afecta el rendimiento de clasificación [10].

```
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
# Dividir datos en características (X) y etiquetas (y)
X = data[['HR', 'PR_interval', 'QRS_duration']]  # Características
y = data['Label']  # Etiquetas: 0 para SB, 1 para AFIB
# Dividir en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
# Entrenar el modelo SVM
svm_model = SVC(kernel='rbf', C=1, gamma='auto')  # Kernel RBF para no linealidad
svm_model.fit(X_train, y_train)
# Realizar predicciones
y_pred = svm_model.predict(X_test)
# Evaluar el modelo
```

```
print("Matriz de Confusión:")
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel("Predicción")
plt.ylabel("Real")
plt.title("Matriz de Confusión - SVM")
plt.show()
print("\nReporte de Clasificación:")
print(classification_report(y_test, y_pred))
```

**Discusión de resultados:** Los SVM son efectivos en problemas de clasificación de alta dimensión, como el análisis de señales ECG, ya que maximizan el margen entre las clases. Sin embargo, pueden ser computacionalmente costosos para grandes conjuntos de datos. Weston y Schölkopf destacan su utilidad en el análisis biomédico, incluyendo la detección de arritmias [8].

**Comparación entre K-means y SVM**

K-means y SVM tienen aplicaciones diferentes. K-means es útil para identificar patrones en datos no etiquetados, mientras que SVM se utiliza para clasificación supervisada. En análisis de señales ECG, K-means puede ser útil para agrupar características no etiquetadas, como ondas P, QRS y T, mientras que SVM es ideal para clasificar patologías específicas. Las bases de datos como MIT-BIH y PhysioNet proporcionan datos estandarizados que pueden utilizarse para comparar ambos enfoques [11], [12].

6. **Consultar por lo menos tres equipos comerciales traen ya incorporados algoritmos de ayuda diagnóstica a partir de señales EKG y discutir brevemente las funcionalidades desde la teoría vista en el curso**

En la actualidad, varios equipos comerciales han integrado algoritmos avanzados para analizar señales ECG y proporcionar asistencia diagnóstica automatizada. A continuación, se describen tres ejemplos destacados:

**1. AliveCor KardiaMobile:** El dispositivo **KardiaMobile** de AliveCor es un monitor portátil de ECG que utiliza algoritmos basados en aprendizaje automático para detectar fibrilación auricular (AFIB), taquicardia, bradicardia y ritmo sinusal normal. Este dispositivo se conecta a un teléfono inteligente y genera un informe de ECG de una derivación en 30 segundos [14]. Desde el punto de vista teórico, los algoritmos incorporados permiten procesar señales en tiempo real, identificar picos R y analizar intervalos clave como PR, QT y QRS. Esto está alineado con los métodos de extracción de características aprendidos en el curso, donde se enfatiza la importancia de segmentar y analizar ondas específicas del ECG.

**2. Apple Watch Series 8:** El **Apple Watch Series 8** incluye un sensor de ECG integrado que genera señales de una derivación. Utiliza un algoritmo basado en aprendizaje supervisado para detectar AFIB y monitorear la frecuencia cardíaca [15]. Los algoritmos del Apple Watch aplican filtrado y segmentación de señales ECG para detectar eventos anormales, similar a las técnicas de procesamiento vistas en el curso. Además, su integración con el entorno iOS permite almacenar y compartir datos con profesionales de la salud para un análisis más detallado.

**3. Holter BTL CardioPoint:** El **BTL CardioPoint Holter** es un dispositivo utilizado en entornos clínicos para monitorear ECG durante 24-48 horas. Integra algoritmos de alta precisión para el análisis de variabilidad de la frecuencia cardíaca (HRV), detección de eventos arrítmicos y clasificación automática de ritmos cardíacos [16]. Este equipo aplica técnicas avanzadas de aprendizaje supervisado y procesamiento de señales, permitiendo a los profesionales obtener informes detallados con segmentación automática de ondas y clasificación basada en patrones predefinidos. Esto está directamente relacionado con la teoría de aprendizaje supervisado y clasificación que se estudió en el curso.

Discusión de funcionalidades

Estos dispositivos comparten características comunes, como:

- **Extracción de características clave:** Frecuencia cardíaca, intervalos PR, QT, QRS, y variabilidad de la frecuencia cardíaca, utilizando algoritmos basados en métodos supervisados o reglas definidas.
- **Clasificación automática:** Emplean modelos de aprendizaje automático o métodos basados en umbrales para detectar ritmos normales y anormales.
- **Interoperabilidad:** La capacidad de compartir datos con aplicaciones móviles o sistemas de gestión clínica para facilitar la interpretación por especialistas.

Desde el enfoque teórico, estas herramientas aplican conceptos de procesamiento de señales como filtrado, segmentación y extracción de características, y combinan estos métodos con algoritmos de clasificación supervisada como Máquinas de Soporte Vectorial (SVM) y Redes Neuronales para mejorar la precisión diagnóstica. Este tipo de dispositivos demuestra cómo los avances en procesamiento de señales y aprendizaje automático pueden integrarse en herramientas comerciales para mejorar la atención médica.

## Conclusiones

- Aunque debido a las circunstancias del kit utilizado y los datos obtenidos durante tiempo reducido a una alta frecuencia de muestreo (500 Hz), sólo se pudo obtener un subconjunto del dataset completo, es posible afirmar que los

resultados obtenidos con las 10 métricas seleccionadas son satisfactorios y presentan la latente posibilidad de generar algoritmos de clasificación de mayor complejidad por medio del Machine Learning y todas sus variantes de mayor complejidad.

- De las tres arquitecturas estudiadas:

| **Característica** | **Pequeña (1 capa)** | **Intermedia (2 capas)** | **Avanzada (3 capas)** |
|---|---|---|---|
| Complejidad | Baja | Media | Alta |
| Riesgo de sobreajuste | Bajo | Moderado | Alto (controlable) |
| Requiere más datos | No | Moderado | Sí (mejor con más datos) |
| Velocidad de entrenamiento | Rápida | Moderada | Lenta |

Es posible concluir que se obtuvo un mejor resultado (accuracy) para la segunda arquitectura, aunque en un escenario hipotético en el que se tengan más datos o mayor cantidad de métricas se deberá evaluar también la tercera arquitectura como un posible candidato.

- En aplicaciones relacionadas a ECG, K-means es útil para explorar datos y descubrir patrones iniciales. En este caso, puede agrupar señales ECG basándose en características como HR, QRS_duration, etc. Por su parte SVM es ideal para clasificar señales etiquetadas, como distinguir entre SB y AFIB, logrando buena precisión con un modelo robusto. Y es trabajo futuro utilizar estas tecnologías para llevar a cabo la clasificación.

- El Machine Learning más allá de ser una moda, es una tecnología que desde años atrás y más aún en la actualidad está revolucionando múltiples campos, no solo de la ciencia, si no también de la vida cotidiana de las personas. Su futuro desarrollo y el subsecuente mejoramiento de las herramientas involucradas serán sin lugar a dudas una de las llaves más importantes para el continuo descubrimiento del funcionamiento de múltiples fenómenos, incluido nuestro cuerpo, sus procesos y señales.

**Referencias**


[1] C. M. Bishop, *Pattern Recognition and Machine Learning*, New York, NY, USA: Springer, 2006. ISBN: 978-0387310732.

[2] J. Han, M. Kamber, y J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed., San Francisco, CA, USA: Morgan Kaufmann, 2012. ISBN: 978-0123814791.

[3] Y. Liu, Z. Li, X. Xue, Y. Zhang, y M. Liu, "Clustering Algorithms: A Comparative Study," *Soft Computing*, vol. 26, no. 3, pp. 1161–1180, 2022. DOI: 10.1007/s00500-021-06073-2.

[4] Scikit-learn, "Clustering — K-means," Scikit-learn Documentation, 2023. [Online]. Available: https://scikit-learn.org/stable/modules/clustering.html#k-means.

[5] A. Ghattas, "The Elbow Method for Selecting K in K-means Clustering," *Towards Data Science*, Aug. 2019. [Online]. Available: https://towardsdatascience.com/the-elbow-method-in-k-means-clustering-5b5e9e9be6b2.

[6] T. Hastie, R. Tibshirani, y J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed., New York, NY, USA: Springer, 2009. ISBN: 978-0387848575.

[7] L. Wang, *Support Vector Machines: Theory and Applications*, 1st ed., Berlin, Germany: Springer, 2005. ISBN: 978-3540243887.

[8] J. Weston y B. Schölkopf, "Support Vector Machines in Biomedical Data Analysis," *Nature Methods*, vol. 2, no. 2, pp. 223–229, 2005. DOI: 10.1038/nmeth.2934.

[9] Scikit-learn, "Support Vector Machines," Scikit-learn Documentation, 2023. [Online]. Available: https://scikit-learn.org/stable/modules/svm.html.

[10] S. Patel, "Support Vector Machines in Plain English," *Towards Data Science*, May 2020. [Online]. Available: https://towardsdatascience.com/support-vector-machines-in-plain-english-f1a5fb4e04d1.

[11] PhysioNet, "PhysioNet: The Research Resource for Complex Physiologic Signals," PhysioNet, 2024. [Online]. Available: https://physionet.org/.

[12] PhysioNet, "MIT-BIH Arrhythmia Database," PhysioNet, 2024. [Online]. Available: https://www.physionet.org/content/mitdb/1.0.0/.

[13] J. L. Johnson et al., "Machine Learning for ECG Analysis," *Journal of Biomedical Informatics*, vol. 91, pp. 103–115, 2019. DOI: 10.1016/j.jbi.2018.12.003.

[14] AliveCor, "KardiaMobile Personal ECG," AliveCor, 2024. [Online]. Available: https://www.alivecor.com/.

[15] Apple Inc., "Apple Watch Series 8," Apple, 2024. [Online]. Available: https://www.apple.com/apple-watch-series-8/.

[16] BTL, "CardioPoint Holter," BTL Industries, 2024. [Online]. Available: https://www.btlnet.com/.