Clasificación de látidos del corazón utilizando ECG's

Roberto Angel Rillo Calva[1 [A01642022]], Jacob Valdenegro Monzón[2,3 [A01640992]], Carlos Dhali Tejeda Tapia[3 [A00344820]], and Enrique Mora Navarro[4 [A01635459]]

Instituto Tecnologico y de Estudios Superiores de Monterrey, México.

Abstract. A lo largo de este documento apuntamos a resolver la problemática que se nos presenta en este reto,

Keywords: Electrocardiograma · Clasificación · Modelos.

1 Introducción

1.1 Electrocardiograma

El electrocardiograma (ECG o EKG) es una herramienta diagnóstica fundamental en la medicina moderna, utilizada para evaluar la actividad eléctrica del corazón.

Un registro de las corrientes eléctricas del corazón es obtenido mediante la colocación de electrodos que contienen un medio conductivo en cada extremidad y en varías localizaciones de la pared del pecho para obtener un ECG de 12 derivaciones. Cada posición específica de los electrodos proporciona un trazado denominado derivación. El propósito de usar 12 derivaciones es para obtener 12 vistas diferentes de la actividad eléctrica en el corazón y por consiguiente un mejor panorama.

El electrocardiograma es importante ya que nos provee de información valiosa acerca del estado cardiaco de un paciente que presente síntomas y señales que sugieran una enfermedad del corazón.

Este examen no invasivo registra los impulsos eléctricos generados durante cada latido del corazón, lo que permite detectar y analizar irregularidades en el ritmo cardíaco, conocidas como arritmias. Las arritmias pueden variar desde benignas hasta potencialmente mortales, y su detección oportuna es crucial para la intervención médica efectiva. [1]

El análisis automático de ECGs ha ganado una relevancia considerable con el avance de la inteligencia artificial y el aprendizaje profundo. Estos métodos prometen mejorar la precisión y eficiencia en la detección de arritmias, permitiendo un diagnóstico más rápido y accesible. En este contexto, se han desarrollado diversas bases de datos para entrenar y evaluar modelos de clasificación de latidos cardíacos, entre las cuales destacan el MIT-BIH Arrhythmia Dataset y la PTB Diagnostic ECG Database. Estas bases de datos ofrecen un amplio conjunto de señales de ECG, que incluyen tanto latidos normales como aquellos afectados por diferentes tipos de arritmias y otros trastornos cardíacos.

El presente trabajo se enfoca en el análisis y modelado de un sistema capaz de detectar arritmias utilizando un dataset compuesto por señales de ECG derivadas de las mencionadas bases de datos. Este dataset ha sido procesado para facilitar su uso en modelos de redes neuronales profundas, que han demostrado un alto potencial en la clasificación de latidos cardíacos.

Este estudio tiene como objetivo construir un modelo de clasificación robusto que pueda identificar de manera precisa las arritmias en señales de ECG, contribuyendo así a la mejora de los sistemas automatizados de diagnóstico cardíaco.

1.2 Clasificación de Ondas en el ECG (Dataset)

El análisis detallado de las ondas en un electrocardiograma (ECG) es clave para la detección precisa de arritmias y otras anomalías cardíacas. En este estudio, utilizamos un dataset que clasifica los latidos cardíacos en cinco categorías principales, cada una representando un tipo específico de ritmo o evento cardíaco. Este dataset es una combinación de dos conjuntos de datos reconocidos: el MIT-BIH Arrhythmia Dataset y la PTB Diagnostic ECG Database. Ambos son ampliamente utilizados en la investigación de la clasificación de latidos cardíacos y han sido preprocesados para facilitar el entrenamiento de redes neuronales profundas.

Composición del Dataset

El MIT-BIH Arrhythmia Dataset aporta un total de 109,446 muestras, mientras que la PTB Diagnostic ECG Database añade otras 14,552 muestras. Todos los datos han sido muestreados a una frecuencia de 125 Hz y ajustados a una dimensión fija de 188 mediante recorte, reducción de muestreo, y padding con ceros cuando es necesario. Este proceso asegura la uniformidad en la longitud de las señales, lo que es crucial para la implementación eficiente de modelos de aprendizaje profundo.

El dataset clasifica las señales de ECG en cinco categorías principales, cada una representada por un código específico:

N (Normal Sinus Rhythm): Esta categoría, codificada como '0', incluye los latidos que presentan un ritmo sinusal normal, caracterizado por una actividad eléctrica regular y coordinada originada en el nodo sinusal.

S (Supraventricular Arrhythmias): Codificada como '1', esta categoría agrupa las arritmias supraventriculares, que son ritmos cardíacos anormales que se originan en las aurículas o en la unión auriculoventricular.

V (Ventricular Arrhythmias): Codificada como '2', esta categoría incluye las arritmias ventriculares, ritmos cardíacos anormales que se originan en los ventrículos y que pueden ser potencialmente mortales si no se tratan adecuadamente.

F (Fusion Beats): Los latidos de fusión, codificados como '3', representan complejos cardíacos que muestran características tanto de latidos normales como anormales. Esta categoría es importante porque su identificación puede indicar la coexistencia de múltiples ritmos cardíacos.

Q (Unknown/Unclassified): Codificada como '4', esta categoría agrupa aquellos latidos o patrones que no pueden ser clasificados con confianza en las

otras categorías. La presencia de estos eventos puede requerir un análisis más profundo o el desarrollo de nuevas categorías de clasificación.

Formato de los Datos

El dataset se encuentra en una serie de archivos CSV, donde cada archivo contiene una matriz. Cada fila de esta matriz representa un ejemplo de latido cardíaco y está formada por una secuencia de valores que describen la forma de onda del ECG para ese latido en particular. El último valor de cada fila corresponde a la categoría a la que pertenece el latido, lo que permite su uso directo en la formación de modelos de clasificación.[2]

2 Metodología

La metodología que utilizaremos se basa en cómo se trabaja en el aprendizaje automatizado, como se observa en la Figura 1. En el aprendizaje automatizado existen 8 pasos, pero debido a que nosotros no seguiremos desarrollando los modelos de clasificación que utilizaremos para resolver la problemática, solo seguiremos hasta el paso número 7 de la Figura 1, "Liberación del modelo para producción".

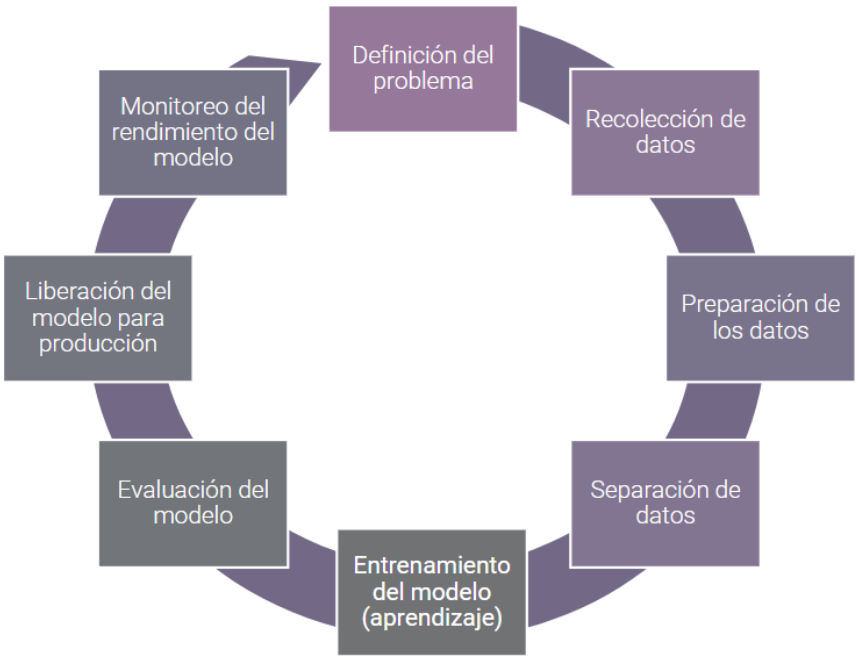

Fig. 1. Metodología de trabajo Aprendizaje Automático.

2.1 Definición del problema

La definición del problema ya se nos había dado desde el inicio de nuestro reto, entonces en este paso simplemente se realizó un poco más de investigación acerca del componente principal de la problemática, que son los electrocardiogramas (ECG). Desarrollamos su concepto, características principales y la importancia de su uso en la introducción de este documento.

2.2 Recolección de datos

La recolección de datos se refiere a todo el proceso de muestreo de datos (*data sampling*), que es la acción de generar los datos con los que se va a trabajar en el problema. Sin embargo, dado que estos *datasets* nos fueron proporcionados desde un inicio, lo único que hicimos fue leer la documentación para entender al menos a un nivel básico la estructura de nuestros datos, para después descargarlos y comenzar a trabajar con ellos.

2.3 Preparación de los datos

Estimación de datos faltantes Desde un inicio realizamos pruebas simples para determinar la estructura inicial de nuestros datos, principalmente gráficándolos. Al ver la forma en que estaban estructuradas las clases de nuestros conjuntos de datos, nos dimos cuenta de que al menos uno de nuestros conjuntos (el MIT-BIH Arrhythmia Database) estaba extremadamente desbalanceado, siendo la clase mayoritaria la que contenía las ondas normales de los pacientes.

Eliminación de datos duplicados

Reducción de ruido

Normalización de datos

Aplicación de cualquier transformación o limpieza En este paso, la transformación que realizamos involucra métodos de balanceo de datos, debido a que las clases en nuestro conjunto de datos están desbalanceadas. Utilizamos submuestreo (*undersampling*) y sobremuestreo (*SMOTE*) en distintas pruebas para encontrar una configuración que mejorara el rendimiento del modelo sin efectos negativos. Seleccionamos el submuestreo para minimizar la clase mayoritaria, y SMOTE para aumentar las muestras de las clases minoritarias.

El submuestreo aleatorio elimina aleatoriamente los puntos de datos de la clase mayoritaria hasta que las clases estén equilibradas. Este método disminuye el tiempo de entrenamiento, pero puede excluir información importante, reduciendo la precisión predictiva del algoritmo.[4]

Por otro lado, SMOTE genera nuevas instancias a partir de casos minoritarios existentes, aumentando las características disponibles para cada clase, haciéndolas más generales. Estas dos técnicas de balanceo fueron las únicas transformaciones que aplicamos a nuestros datos originales.[3]

Extracción de características

Eliminación de variables correlacionadas

2.4 Separación de los datos

El conjunto de datos de MIT-BIH ya estaba separado en conjuntos de entrenamiento y prueba. Además, utilizamos validación cruzada, que separa el conjunto de entrenamiento en subconjuntos de entrenamiento y validación. Al final, evaluamos con nuestro conjunto de prueba. Para este paso, fue importante la utilización de validación cruzada.

Para el conjunto de datos de PTB el cual contaba con dos archivos, uno normal y uno anormal, decidimos juntarlos de forma aleatoria para posteriormente dividirlo en un 80

2.5 Entrenamiento del modelo

Entrenamiento preliminar con validación cruzada Una vez conocida la estructura de nuestros datos, comenzamos a probar modelos de clasificación sin transformación para observar el comportamiento de los datos desbalanceados y comparar con métodos de balanceo. Probamos varios modelos simultáneamente para filtrar rendimiento.

La validación cruzada fue crucial, ya que permite evaluar el modelo utilizando todos los datos disponibles, dividiendo aleatoriamente el conjunto en k particiones o pliegues (*k-fold cross validation*) como podemos ver en la figura 2. Utilizamos *StratifiedKFold*, que mantiene la distribución de las clases en cada pliegue, ayudando a trabajar con clases desbalanceadas.

Evaluación del rendimiento Para evaluar el rendimiento de los modelos, utilizamos la herramienta `classification_report` de la librería `sklearn`, que nos permitía ver métricas clave, como:

Sensibilidad: Cantidad de muestras reales de una clase que están siendo clasificadas correctamente en esa clase. Es importante para evitar clasificar incorrectamente a pacientes enfermos.

Como un ejemplo, en la figura 3 se puede observar una matriz de confusión la cuál nos permite visualizar gráficamente hacía cómo se están comportando las muestras de nuestras clases. El recall de la clase 3 sería el porcentaje de muestras reales del total que si están clasificadas en la clase 3, en este caso son 379. Esta métrica es importante para nosotros ya que al haber más recall se reduce el

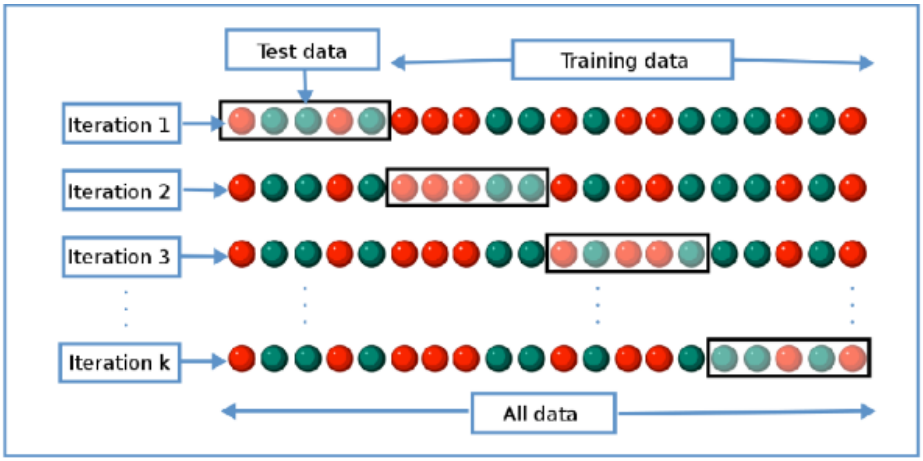

Fig. 2. Validación cruzada en k-pliegues.

riesgo de que las muestras reales sean clasificadas como de otra clase, esto es especialmente importante en nuestro caso ya que queremos evitar a toda costa que los pacientes que si tengan una enfermedad (osea las muestras de la clase 1 a 4) sean clasificadas como pacientes normales ya que esto podría ocasionar riesgos fatales para estas personas.

Precisión: Porcentaje de predicciones correctas en cada clase. Nos concentramos en la precisión de la clase 0 para evitar clasificar pacientes normales como enfermos.

En la figura 4 podemos observar todas las predicciones que se realizaron para la clase 0, que son las subrayadas verticalmente, la precisión en este caso sería el porcentaje de las que si se clasificaron en la clase 0, o sea 72,349 de un total de 74,312, 97

F1-score: Promedio entre precisión y sensibilidad para cada clase.

Precisión promediada: Promedio de los F1-scores de todas las clases. Suele ser utilizado para indicar la eficacia general de un modelo, pero tenemos que tener cuidado ya que este valor puede ser bastante alto a pesar de tener baja sensibilidad en algunas clases de nuestro modelo, ya que todo termina siendo promediado.

Tras observar estas métricas, seleccionamos los modelos más prometedores para probar con balanceo de datos.

Entrenamiento de modelos con balanceo Probamos diferentes configuraciones de submuestreo y SMOTE, tomando decisiones basadas en las métricas obtenidas. Consideramos factores como el tiempo de entrenamiento y el riesgo de sobreajuste. En todos los casos, el balanceo solo se aplicó al conjunto de entrenamiento.

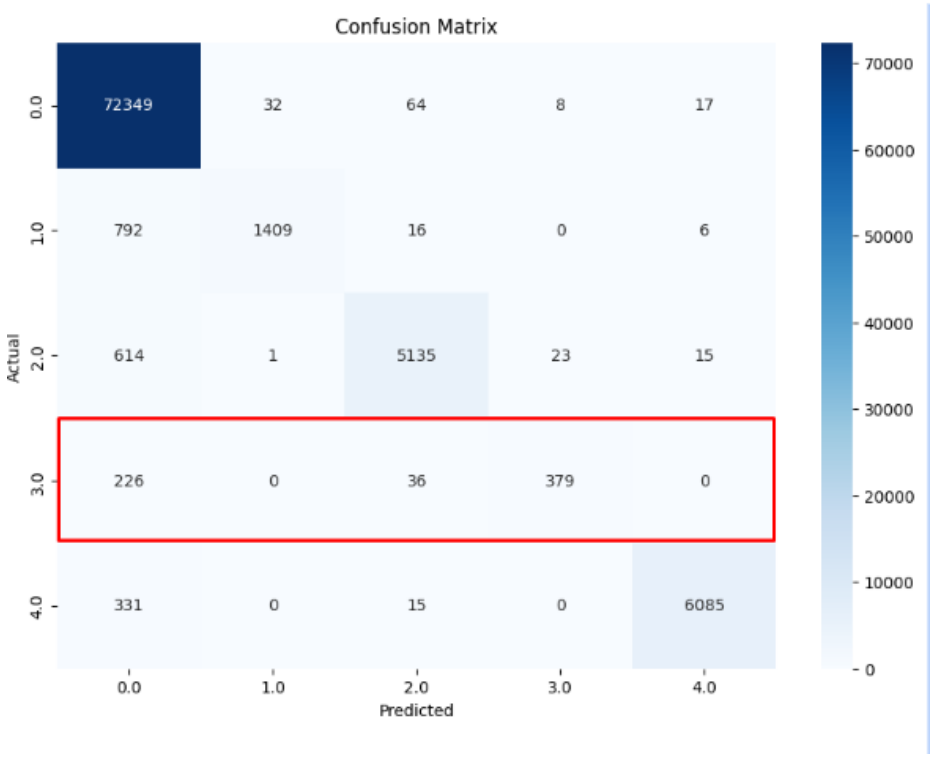

Fig. 3. Ejemplo de sensibilidad en matriz de confusión.

Algo que tomamos en cuenta a la hora de probar los modelos con balanceo en la validación cruzada es que el submuestreo y sobremuestreo se realizan únicamente sobre el conjunto de entrenamiento que se pasa al algoritmo de optimización, incluyendo en validación cruzada. No se debe considerar que la muestra submuestreada o sobremuestreada es un nuevo conjunto de datos para llevar a cabo todo el análisis.

Ajuste de hiperparámetros Luego de obtener resultados preliminares, ajustamos los hiperparámetros de los modelos seleccionados usando *RandomSearch* y *GridSearch*. Nos limitamos a configuraciones que no tomaran demasiado tiempo de ejecución. Finalmente, evaluamos el modelo optimizado con el conjunto de prueba.

2.6 Evaluación del modelo candidato

Una vez con los mejores hiperparámetros, evaluamos el modelo en el conjunto de prueba para determinar su rendimiento final. Seleccionamos el modelo que mejor equilibraba precisión y *recall* en las clases que nos interesaban mejorar.

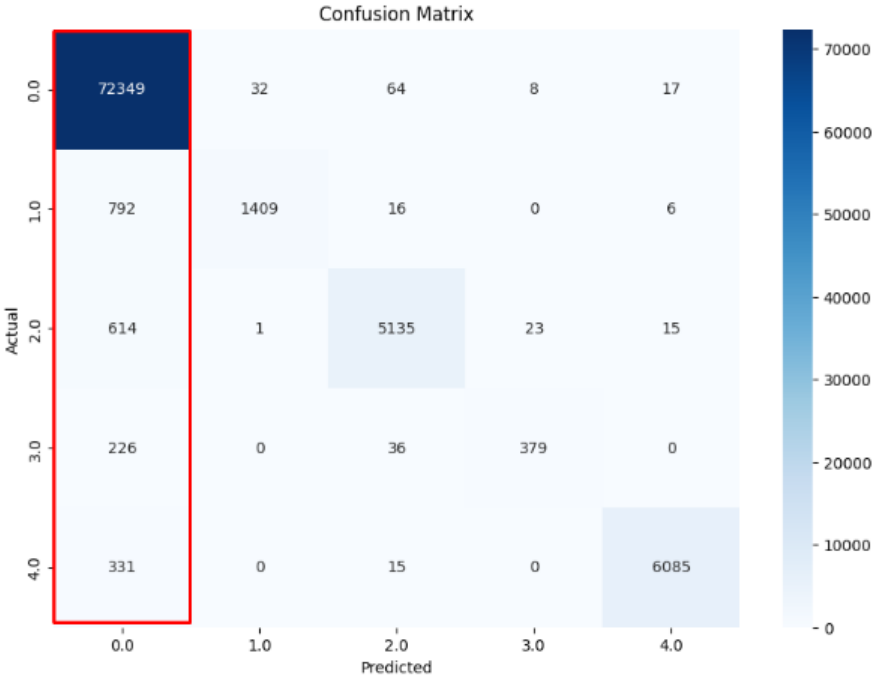

Fig. 4. Ejemplo de precisión en matriz de confusión.

3 Experimentos

3.1 Experimentos en conjunto de datos MITBIH

Lo primero que hicimos fue seleccionar varios modelos arbitrariamente para poder entrenarlos sin ningún tipo de balanceo en nuestro conjunto de datos. Para este conjunto de datos seleccionamos 5 modelos y los probamos:

- RandomForest
- BalancedRandomForest
- WeightBalanced RandomForest
- MLP
- K-Neighbors

Una vez hechas estas pruebas, definimos qué modelos usaríamos en pruebas futuras y nos enfocamos en probar distintos balanceos. Probamos los siguientes tipos de balanceo:

1. Solo submuestreo de la clase mayoritaria, dejándola en 20,000 muestras.
2. Submuestreo de la clase mayoritaria hasta 20,000 y sobremuestreo SMOTE en todas las demás clases, dejando todas las clases con 20,000 muestras cada una.

3. Submuestreo de 20,000 muestras en la clase mayoritaria y sobremuestreo SMOTE en las clases 1, 2 y 3.

Modelos probados:

- RandomForest
- BalancedRandomForest
- WeightBalanced RandomForest

Prueba extra: Decidimos darle otra oportunidad a `KNeighbors`, pero con estandarización de los datos, ya que leímos que modelos como `KNeighbors` podían beneficiarse de esto. Entonces probamos `KNeighbors` con estandarización y ajuste de hiperparámetros, pero aun así no obtuvimos resultados relevantes.

3.2 Pruebas de hiperparámetros

Después de las pruebas de balanceo, pasamos a pruebas de hiperparámetros utilizando los métodos de `RandomSearch` y `GridSearch`. Con `GridSearch` probamos las siguientes configuraciones:

- `BalancedRandomForest` con los datos originales para evaluar su rendimiento teniendo en cuenta que es un modelo con balanceo interno.
- `BalancedRandomForest` con los datos balanceados (20,000 muestras en cada clase).
- `RandomForest` con `class_weight` en datos balanceados (20,000 muestras en la clase mayoritaria, 6,000 en las demás clases).

Hiperparámetros utilizados:

```
param_grid_rf = {
    'model__n_estimators': [100, 200, 500],  # Número de árboles
    'model__max_depth': [10, 20, None],      # Profundidad máxima de los árboles
    'model__min_samples_split': [2, 5, 10],  # Mínimo de muestras para dividir un nodo
}
```

Realizamos pruebas con diferentes configuraciones del hiperparámetro `model__n_estimators` y nos dimos cuenta de que el modelo siempre tendía a elegir el número mayor. Decidimos dejarlo de esta manera, ya que aumentar a valores como 1,000 o más consumía mucho tiempo.

3.3 Experimentos en conjunto de datos PTBDB

Para este conjunto de pruebas utilizamos únicamente 2 modelos:

- RandomForest
- SVC

Una vez realizadas estas pruebas, decidimos buscar hiperparametros para mejorar alguno de estos modelos utilizando únicamente RandomSearch.

```
param_dist = {
    'n_estimators': np.arange(100, 301, 50),   # Mejor número de árboles
    'max_depth': [None, 10, 20, 30],           #Límite de profundidad
    'min_samples_split': np.arange(2, 11, 3),  #Mínimo de muestras para dividir un nodo
    'min_samples_leaf': [1, 2, 4],             # Mínimo de muestras en cada hoja
    'max_features': ['sqrt', 'log2', None],    # Máximo número de características para divi
    'bootstrap': [True, False]
}
```

4 Resultados

Los diferentes resultados para las diferentes etapas ya mencionadas de los experimentos realizados tanto para el dataset de MIT como para el de PTB acorde a la metodología elegida, son los siguientes

4.1 MITBIH Dataset

Random Forest El reporte de clasificación de la librería scikit-learn y la matriz de confusión para el modelo de clasificación de Random Forest del dataset de MIT con un submuestreo de 20000 para la clase mayoritaría y sobremuestreo SMOTE para la clase 1, 2 y 3 en la validación cruzada por cada pliegue, junto con los mejores hiperparámetros seleccionados por GridSearch para los datos de testing es el siguiente:

	Precision	**Recall**	**F1-score**	**Support**
0.0	0.98	0.99	0.99	18117
1.0	0.83	0.76	0.79	556
2.0	0.96	0.92	0.94	1448
3.0	0.72	0.79	0.75	162
4.0	0.99	0.96	0.97	1608
Accuracy			0.98	21891
Macro avg	0.90	0.88	0.89	21891
Weighted avg	0.98	0.98	0.98	21891

Table 1. Resultados de precisión, recall, F1-score y support para cada clase

Balanced Random Forest El reporte de clasificación de la librería scikit-learn y la matriz de confusión para el modelo de clasificación de Balanced Random Forest del dataset de MIT que ofrece un balanceo interno y aparte un submuestreo

Fig. 5. Matriz de confusión para las clases reales y predichas

de 20000 para la clase mayoritaría y sobremuestreo SMOTE para la clase 1, 2 y 3 en la validación cruzada por cada pliegue, junto con los mejores hiperparámetros seleccionados por GridSearch para los datos de testing es el siguiente:

	Precision	**Recall**	**F1-score**	**Support**
0.0	0.99	0.96	0.98	18117
1.0	0.58	0.82	0.68	556
2.0	0.87	0.95	0.91	1448
3.0	0.48	0.83	0.61	162
4.0	0.96	0.97	0.96	1608
Accuracy			0.96	21891
Macro avg	0.78	0.91	0.83	21891
Weighted avg	0.97	0.96	0.96	21891

Table 2. Resultados del Balanced Random Forest: precisión, recall, F1-score y soporte para cada clase

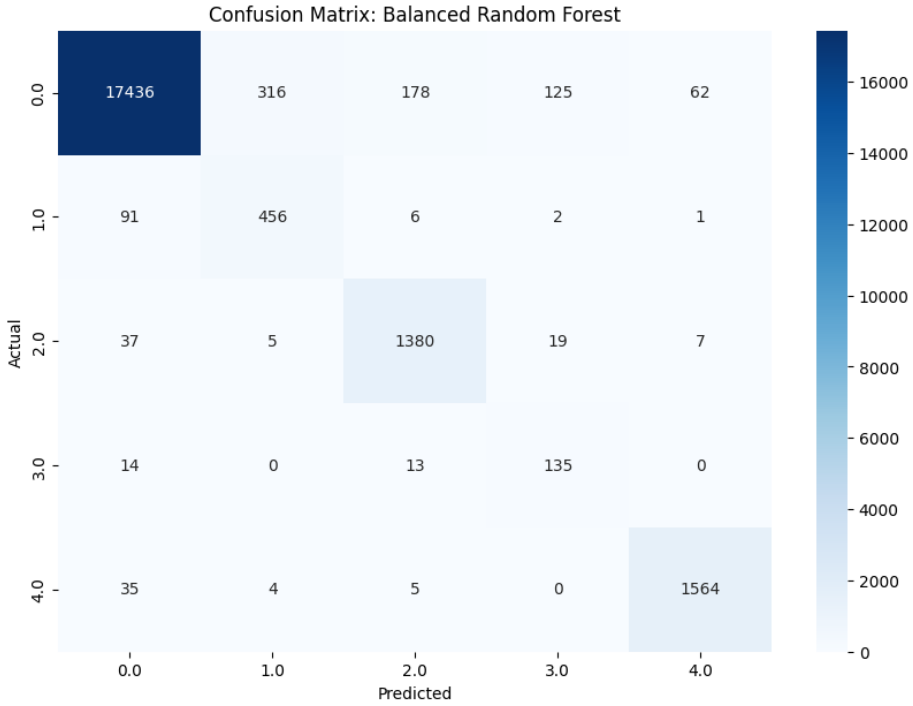

Fig. 6. Matriz de confusión para las clases reales y predichas pero del modelo Balanced Random Forest

4.2 PTB Dataset

Random Forest El reporte de clasificación de la librería scikit-learn y la matriz de confusión para el modelo de clasificación de Random Forest del dataset de PTB junto con los mejores hiperparámetros seleccionados por Random Search para los datos de testing es el siguiente:

	Precision	**Recall**	**F1-score**	**Support**
0.0	0.97	0.92	0.95	809
1.0	0.97	0.99	0.98	2102
Accuracy			0.97	2911
Macro avg	0.97	0.96	0.96	2911
Weighted avg	0.97	0.97	0.97	2911

Table 3. Resultados de precisión, recall, F1-score y soporte

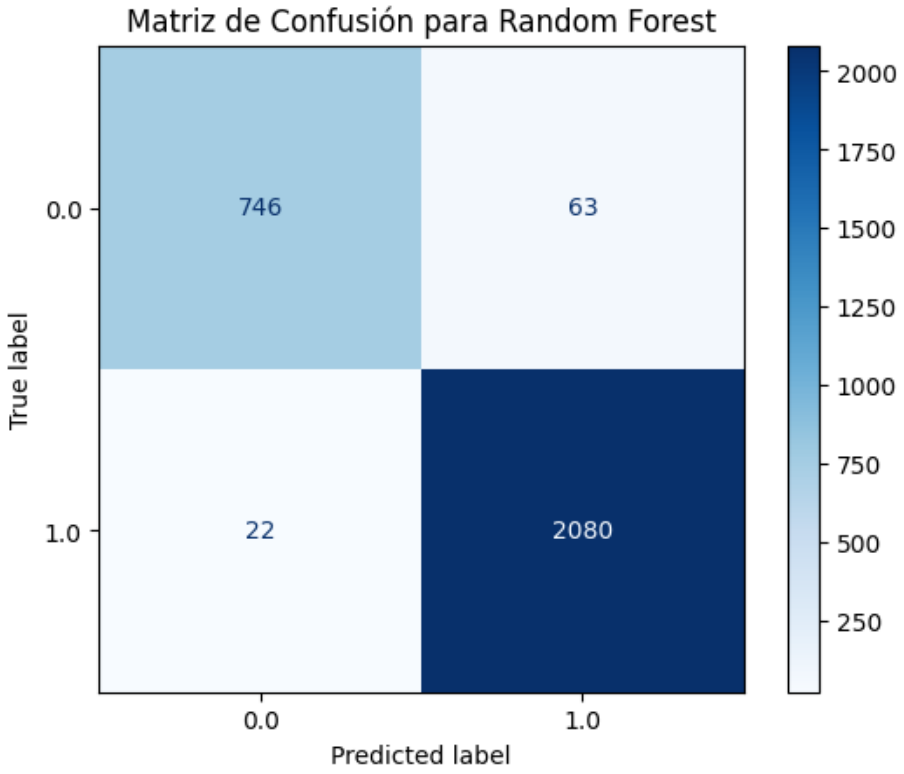

Fig. 7. Matriz de confusión para las clases reales y predichas pero del modelo Balanced Random Forest

SVC El reporte de clasificación de la librería scikit-learn y la matriz de confusión para el modelo de clasificación de SVC del dataset de PTB para los datos de testing es el siguiente:

	Precision	**Recall**	**F1-score**	**Support**
0.0	0.90	0.82	0.86	809
1.0	0.93	0.96	0.95	2102
Accuracy			0.93	2911
Macro avg	0.92	0.89	0.90	2911
Weighted avg	0.92	0.93	0.92	2911

Table 4. Resultados de precisión, recall, F1-score y soporte

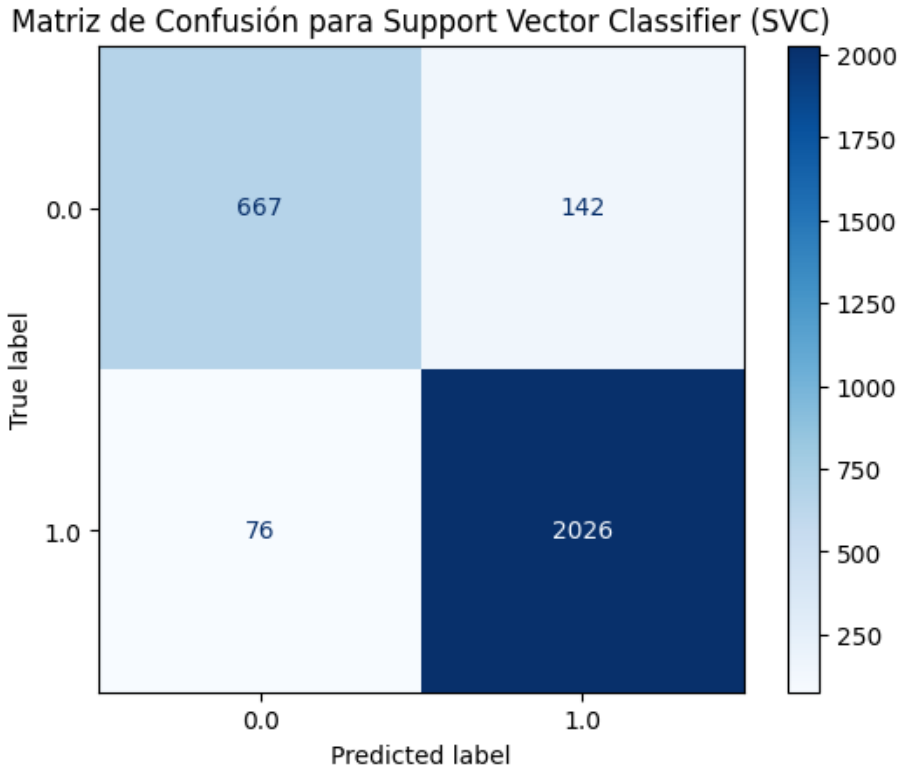

Fig. 8. Matriz de confusión para las clases reales y predichas pero del modelo SVC para PTB

5 Conclusiones

Después de realizar diversas pruebas con el dataset de MIT-BIH utilizando tanto el Balanced Random Forest como el Random Forest con hiperparámetros óptimos, hemos decidido seleccionar el modelo de Random Forest para nuestra solución final. Esta decisión se basa en una evaluación exhaustiva de los resultados de los modelos, considerando tanto la matriz de confusión como el reporte de clasificación.

El Balanced Random Forest mostró resultados prometedores y mejores métricas en cuanto a las recalls de las clases minoritarias. Sin embargo, decidimos no utilizarlo debido al riesgo de sobreajuste (overfitting) asociado con su estrategia de balanceo redundante. El Balanced Random Forest realiza un balanceo interno adicional al balanceo externo mediante undersampling y SMOTE en cada pliegue durante la validación cruzada. Esta doble técnica de balanceo puede inducir redundancia y aumentar el riesgo de sobreajuste, afectando negativamente la capacidad del modelo para generalizar a datos nuevos.

Además, priorizamos obtener altas tasas de recall para las clases minoritarias 1, 2, 3 y 4, en lugar de enfocarnos únicamente en la precisión general del modelo. La clase 0 (Normal Sinus Rhythm) es la clase mayoritaria en nuestro dataset, y todos los modelos mostraron una precisión superior al 97% esta clase. Sin embargo, el riesgo de obtener falsos positivos en las clases minoritarias es significativo, especialmente debido al marcado desbalance entre las clases.

El Random Forest, al evitar el balanceo interno redundante, demuestra una mayor robustez y una mejor capacidad para manejar el desbalance de clases sin comprometer la capacidad de generalización. Además, su desempeño en la clasificación de las clases minoritarias fue más consistente y confiable, alineándose con nuestra necesidad de minimizar el riesgo de falsos positivos en casos clínicos críticos.

De igual forma, para los datos de PTB, el Random Forest con hiperparámetros optimizados sigue siendo el modelo más efectivo. Logra un mayor recall y precisión tanto para la clase normal como para la clase anormal. Dado que el objetivo principal es reducir los falsos negativos para la clase de anormales (recall de la clase 1), Random Forest ofrece una mejor capacidad para detectar estos casos, lo cual es esencial en este contexto médico. Además, mantiene un buen equilibrio en el desempeño para ambas clases, alcanzando un accuracy del 97% en el conjunto de prueba.

En conclusión, Random Forest es la mejor opción en ambos escenarios. En el caso del MIT dataset, ofrece una mayor robustez y capacidad de generalización sin comprometer la detección de clases minoritarias. Para el PTB, su capacidad para detectar correctamente ECG anormales con un recall muy alto (99%) y sin comprometer la precisión lo convierte en una solución ideal para este problema médico.

6 Conclusiones Personales

6.1 Enrique Mora

Este proyecto me ayudo a mejorar mi capacidad para interpretar datos y entender que la evaluación y la implementación de modelos no dependen únicamente de la precisión, sino del contexto de la base de datos y de una lectura profunda de los resultados. Aprendí que métricas como el recall y herramientas visuales como las gráficas y la matriz de confusión son esenciales para tomar decisiones informadas y reflejar con mayor claridad la realidad de los datos. Este enfoque integral me ha enseñado que una comprensión detallada y un análisis multifacético son fundamentales para extraer conclusiones significativas y precisas.

6.2 Carlos Dhali

Todo el proceso de este reto a pesar de su naturaleza retadora me pareció una increíble oportunidad de aplicar los aprendizajes que nos fueron inculcando a lo largo de estas 6 semanas. En un inicio sentía que no teníamos una visión clara y

que no se lograría, pero conforme fueron pasando los días, fuimos haciendo más pruebas, empecé a sentir que de verdad estaba aprendiendo e involucrándome en el proceso de este reto. El mayor reto para mí fue el tomar decisiones en cuanto a las pruebas que estábamos haciendo, ¿sigo corriendo esta prueba que ya lleva más de una hora?, ¿será esto que estoy haciendo la forma correcta de hacerlo?, ¿debería de seguir por este camino o mejor probar otros modelos o otros balanceos?, a mi parecer este componente de libertad tan grande en la toma de decisiones es un reto muy grande, sobre todo por la naturaleza de la problemática que te hace pagar con mucho tiempo si tomas una decisión incorrecta. Creo que con lo que más me quedo de todo esto es que no siempre se trata de encontrar los mejores resultados si no de seguir un proceso y cuando obtengas resultados saber explicar el como llegaste ahí, porque obtuviste el resultado que obtuviste y que cosas pudiste haber mejorado.

6.3 Jacob Valdenegro

Gracias a este proyecto pude entender mas acerca de como abordar proyectos que requieran interpretar datos, aprendí que la forma de buscar un modelo que se adapte a las necesidades del problema es haciendo muchas pruebas, ya que aunque te de un resultado terrible, vas entendiendo que hacer y que no, también aprendí a interpretar las métricas como la precisión y el recall, o las matrices de confusión, llegando así a conclusiones mas precisas.

6.4 Roberto Rillo

A lo largo de este reto de 6 semanas, adquirí una comprensión sólida de los fundamentos de la inteligencia artificial, y más específicamente, de machine learning. No solo aprendí a implementar modelos de clasificación y regresión en Python, sino que también desarrollé la capacidad de interpretar tanto los datos como los resultados de manera crítica. Uno de los aprendizajes más valiosos fue cómo manejar adecuadamente los datasets, incluyendo técnicas para balancearlos cuando es necesario, y sobre todo, entender el contexto del problema al que se aplican estos modelos.

Este enfoque contextual resultó ser clave para lograr una solución precisa. No se trata solo de obtener métricas altas en términos numéricos, sino de comprender e interpretar qué es lo verdaderamente importante en cada escenario y qué aspectos pueden sacrificarse para minimizar riesgos o evitar resultados costosos, como los falsos negativos o positivos en problemas médicos críticos. En conclusión, aprendí que la elección de un modelo no debe centrarse únicamente en la precisión global, sino también en cómo responde a las particularidades del reto, como el desbalance de clases o la importancia de ciertas métricas como el recall.

References

1. Albert Heuer. *Wilkins' Clinical Assessment in Respiratory Care - E-Book: Wilkins' Clinical Assessment in Respiratory Care - E-Book.* Elsevier Health Sciences, 2017, 8 edition, 10 2017.
2. Kaggle.
3. Likebupt. SMOTE - Azure Machine Learning, 9 2024.
4. Tarid Wongvorachan, Surina He, and Okan Bulut. A comparison of undersampling, oversampling, and smote methods for dealing with imbalanced classification in educational data mining. *Information*, 14(1):54, 2023.