PROYECTO - CLASIFICACIÓN DE ECG Y EXTRACCIÓN DE CARACTERÍSTICAS SIMPLES

Este proyecto pretende clasificar las señales de electrocardiograma humano (ECG) mediante algoritmos de Machine Learning (ML). Pasaremos por el flujo de trabajo de extracción de características (simples) y entrenaremos varios modelos ML que pueden reconocer los siguientes ritmos cardíacos:

- Ritmo Sinusal Normal (N)
- Arritmia (A)

Los datos de ECG originales provienen del PhysioNet 2017 Challenge, que está disponible en https://physionet.org/challenge/2017/.

Cargar y Preprocesar Datos

El archivo `PhysionetData.mat` contiene las variables:

- `Signals`: arreglo de celdas que contiene señales de ECG
- `Labels`: arreglo categórico que contiene las etiquetas reales correspondientes

Tarea: Cargue el archivo `PhysionetData.mat` y visualice las primeras 5 filas de las variables `Signals` y `Labels`. Utilice el comando `summary` para observar la cantidad de señales con Ritmo Sinusal Normal (N) y con Arritmia (A).

Como podemos ver en la vista previa, la longitud de la señal varía (ver variable `Signals`).

Tarea: Cree un histograma que muestre la longitud de las señales. Para ello use la función `plotSignalLengths` (proporcionada con este proyecto).

La mayoría de las señales de ECG tienen una longitud de 9000 muestras, pero existe cierta variabilidad.

Tarea: Visualice un segmento de una señal de cada clase. Para ello use la función `visualizeECGSignals` (proporcionada con este proyecto).

Aquí hay dos características comunes de las señales con Arritmia (A):

- Espaciados a intervalos irregulares.
- Carece de una onda P, que pulsa antes del complejo QRS en latidos cardíacos normales.

Dado que las señales de ECG tienen diferentes longitudes, primero debemos rellenarlas y truncarlas para que tengan una longitud de 9000 muestras.

La función `segmentSignals` (proporcionada con este proyecto) ignora las señales con menos de 9000 muestras. Si la señal tiene más de 9000 muestras, `segmentSignals` la divide en tantos segmentos de 9000 muestras como sea posible e ignora las muestras restantes.

Por ejemplo, una señal con 18500 muestras se convierte en dos señales de 9000 muestras y las 500 muestras restantes se ignoran.

Tarea: Utilice la función `segmentSignals` para redimensionar las variables `Signals` y `Labels`. Con la función `plotSignalLengths` cree un nuevo histograma para verificar la longitud de las señales.

Extraer Características

Los modelos ML usan características (o predictores) como entrada (llamadas variables dependientes) y devuelven un valor predicho como salida (llamada variable independiente). Las características pueden ser medidas externas o cualquier otro valor relacionado con la salida. Este último puede verse como características internas que se pueden generar mediante la transformación de los datos originales.

Extraer las mejores características adecuadas requiere conocimientos de dominio. Las medidas estadísticas de los datos pueden ser un buen comienzo para extraer características. Las mejores características son aquellas que muestran una gran diferencia entre las diferentes clases.

Primero, echemos un vistazo a la distribución de las señales de ambas categorías trazando un histograma de una señal de cada clase. La función `plotBothHistograms` (proporcionada con este proyecto) también calcula y muestra la mediana, la desviación estándar, la asimetría, y la curtosis de ambas señales.

Tarea: Separe las señales con Ritmo Sinusal Normal (N) de las señales con Arritmia (A) y almacene la información en variables separadas. Con la función `plotBothHistograms` cree un nuevo histograma para ver las medidas estadísticas de cada grupo de señales.

A juzgar por las variables estadísticas, se puede observar que ambas señales (N y A) presentan características muy diferentes. Por esta razón, las usaremos como características. También agregaremos la desviación media absoluta, cuantiles 25 y 75, y el rango intercuartílico de la señal. La función `extractFeatures` (proporcionada con este proyecto) calcula características para todo el conjunto de datos.

Tarea: Utilice la función `extractFeatures` para crear una tabla que contenga las 10 medidas estadísticas de las señales.

Nota: debido a la gran cantidad de datos, esto puede demorar varios segundos.

Preparar Datos para el Entrenamiento

Para poder evaluar el rendimiento de los modelos, dividimos nuestro conjunto de datos en dos subconjuntos: un conjunto de entrenamiento y un conjunto de prueba. El conjunto de entrenamiento se utilizará para entrenar el/los modelo(s), mientras que el conjunto de prueba se utilizará después del entrenamiento para evaluar el rendimiento en datos nuevos (no vistos).

Se recomienda utilizar entre el 60 y el 90% de todos los datos para entrenamiento y el resto para pruebas. Una forma de dividir los datos es usar el comando `cvpartition`.

Tarea: Utilice el comando `cvpartition` para dividir el 80% de los datos para el entrenamiento y el resto para las pruebas. Almacene estos datos en las variables `TrainData` y `TestData`, respectivamente. Use el comando `summary` para observar la cantidad de señales con Ritmo Sinusal Normal (N) y con Arritmia (A) de `TrainData` y `TestData`.

Entrenar usando Classification Learner

Uno de los mayores desafíos en ML es que existen muchos algoritmos ML diferentes que podrían aplicarse al mismo problema, y no hay forma de saber de antemano cuál funcionará mejor. La aplicación Classification Learner nos brinda una manera fácil de probar muchas técnicas de modelado diferentes muy rápidamente.

Para abrir la aplicación Classification Learner, puede ejecutar el comando

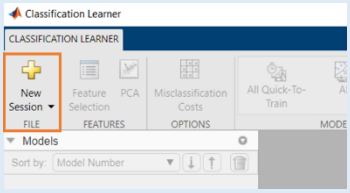
```
classificationLearner
```

o ir a APPS y hacer clic en 

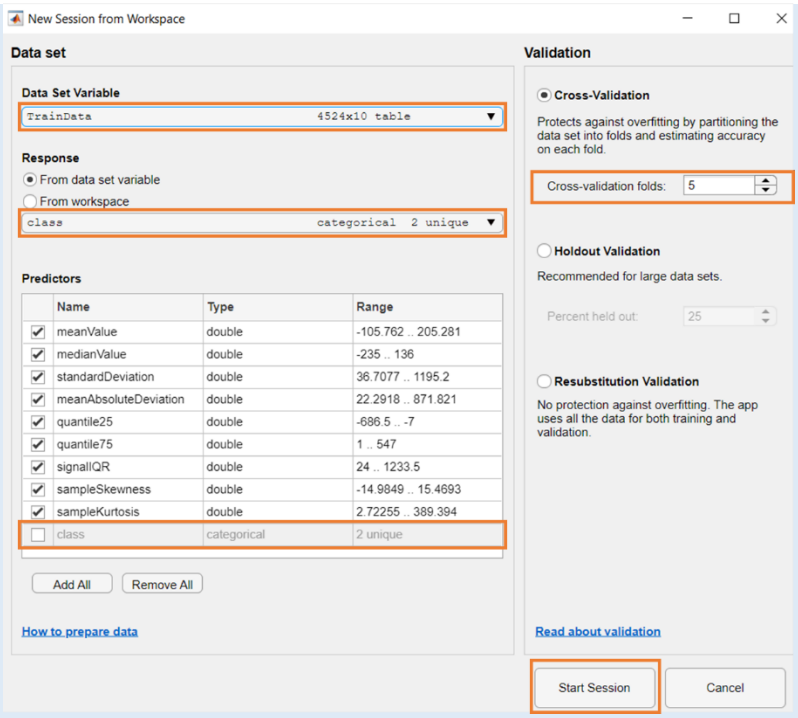
.

Tarea:

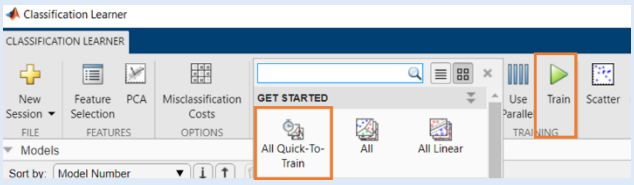
- Inicie una `New Session`



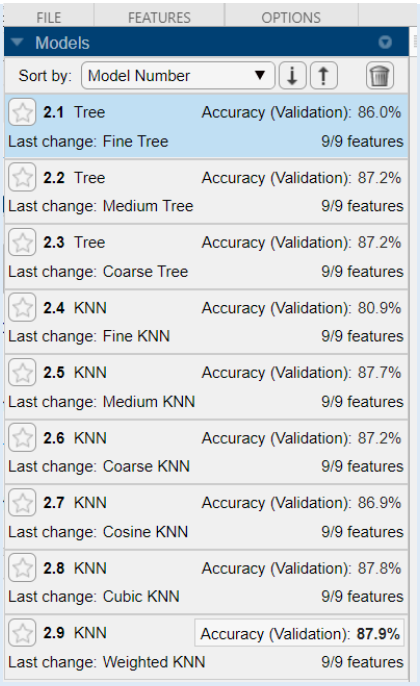
- Elija la variable `TrainData` del espacio de trabajo como sus datos. Asegúrese de que `class` esté seleccionada como `Response` y que todas las casillas de `Predictors` estén marcadas (excepto `class`). Mantenga la sección `Validation` como está (`Cross-Validation` con `5 folds`) y haga clic en `Start Session`.



- En `Model Type`, elija `All Quick-To-Train` y comience a entrenar haciendo clic en `Run`. El entrenamiento puede tardar varios segundos.



- En la ventana `Models`, debería ver el progreso del entrenamiento y, cuando termine, la precisión final de cada modelo. La mayor precisión está resaltada en negrita. Sus resultados deberían verse similares a esto:



- Explore las matrices de confusión de cada modelo haciendo clic en el modelo individual y en `Confusion Matrix`.

Abordar el Desequilibrio de Datos

La mayoría de los modelos entrenados tienen una tasa de falsos negativos muy alta para la señal con Arritmia (A). Esto se debe al desequilibrio en las clases (recuerde que el conjunto de entrenamiento contiene 582 señales con Arritmia (A) y 3942 señales Ritmo Sinusal Normal (N)) lo que hace que el modelo esté sesgado hacia la clase principal (en nuestro caso, la señal normal).

Hay varias formas de manejar clases desequilibradas como, por ejemplo, aplicando las técnicas de sobremuestreo o submuestreo. Otra forma de reducir la clasificación errónea es utilizar modelos sensibles a los costos que penalizan a la clase mayoritaria. La aplicación Classification Learner ofrece una opción para especificar la matriz de costos de clasificación errónea. La búsqueda de los mejores coeficientes de costos de clasificación errónea a menudo se realiza manualmente mediante prueba y error.

Tarea: En `Model Type`, elija `All Quick-To-Train models` y modifique `Misclassification Costs`. Las diagonales de la matriz de clasificación errónea deben ser cero.

Puede encontrar que la tasa de falsos negativos para la señal con Arritmia (A) se reducirá, sin embargo, aumentará para la señal normal. Es muy probable que la precisión general también disminuya (a menos que encuentre los costos correctos de clasificación errónea).

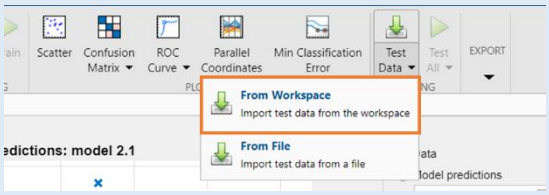
Opcional: intente equilibrar los datos. Puede encontrar esto y esto útil.

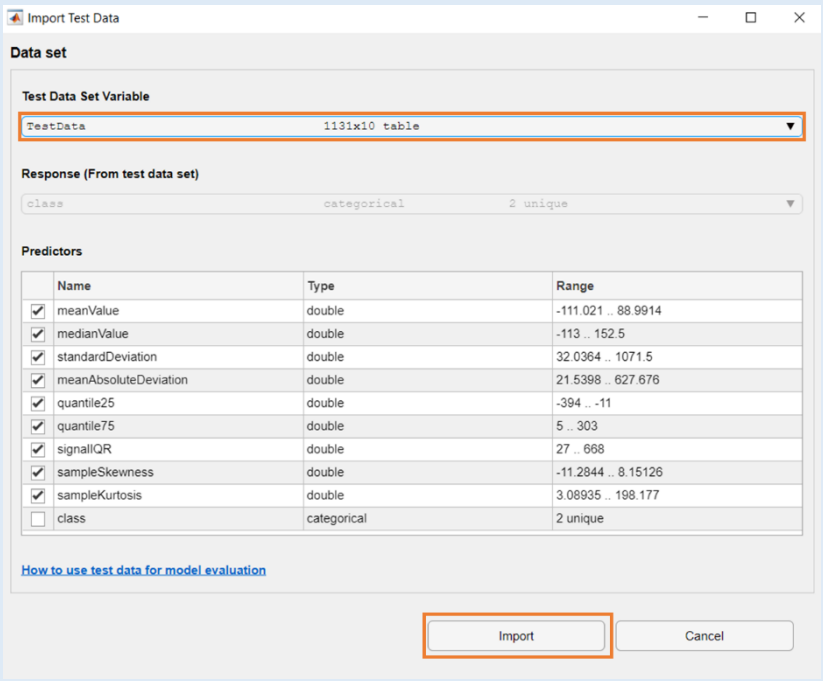
Evaluar el Modelo Entrenado

Puede evaluar el rendimiento de cada modelo en el nuevo conjunto de datos (no visto).

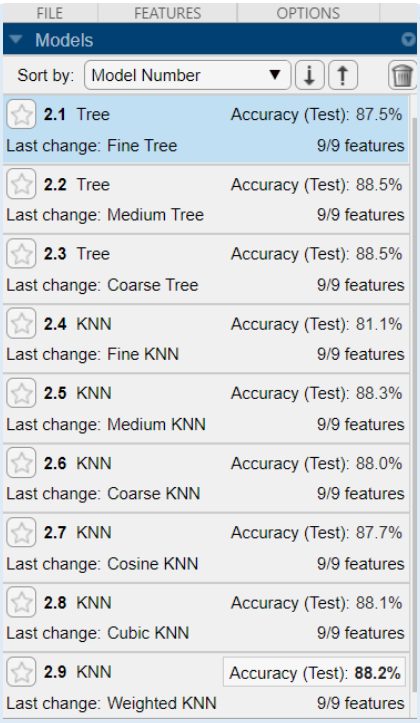
Tarea:

- Cargue el conjunto de datos de prueba haciendo clic en el menú desplegable `Test Data` y seleccionando `From Workspace` (seleccione `TestData` como entrada) y haga clic en `Import`.





- Elija un modelo cuyo rendimiento desee evaluar haciendo clic en el `Star Button` (Agregar a favoritos) y haga clic en `Test Selected` en el menú desplegable `Test All`. Alternativamente, no seleccione ningún modelo y haga clic en `Test All` (la precisión en la ventana `Model` cambiará a `Accuracy(Test)`. Sus resultados deberían verse similares a esto:



- Puede utilizar la matriz de confusión para visualizar dónde se equivocó el modelo en los datos de prueba.

Como era de esperar, los modelos funcionan muy bien en señales normales (N) y clasifican erróneamente la mayoría de las señales con Arritmia (A). Esto se debe al conjunto de datos desequilibrado.

Nota: Alternativamente, puede exportar su modelo entrenado o generar una función para su uso posterior.

Resumen y Próximos Pasos

En este proyecto, creamos modelos ML que identifican la arritmia cardíaca a partir de formas de onda de ECG. Extrajimos características básicas aplicando medidas estadísticas, entrenamos interactivamente algunos modelos con Classification Learner y comparamos su desempeño.

Claramente, con la mayor precisión aún por debajo del 90%, no hemos resuelto satisfactoriamente este problema. A continuación, se indican algunos pasos adicionales que puede intentar para mejorar la precisión:

- Mejores características: al aprovechar el conocimiento del dominio médico y el procesamiento de señales, puede extraer funciones más potentes. La segunda parte del proyecto lo guiará a través de la extracción de características que producen modelos de alta precisión.
- Abordar el desequilibrio de datos: sus modelos fácilmente clasificaron erróneamente las formas de onda de ECG anormales como normales porque hay casi 10 veces más casos de ECG normal

en el conjunto de datos, pero esa clasificación errónea tiene serias consecuencias prácticas (alguien con una afección cardíaca no recibiría tratamiento).

- Ajuste de hiperparámetros: cada modelo tiene parámetros que deben configurarse correctamente para lograr un rendimiento óptimo. Puede hacer que MATLAB ajuste automáticamente esos hiperparámetros eligiendo un modelo "Optimizable" de la galería en Classification Learner. Esto producirá una pequeña mejora en la precisión, típicamente 0.5% o menos, ya que la configuración de los parámetros predeterminados en MATLAB se elige cuidadosamente.

Archivos requeridos:

`PhysionetData.mat`

`extractFeatures.m`

`plotBothHistograms.m`

`plotSignalLengths.m`

`segmentSignals.m`

`visualizeECGSignals.m`