PROYECTO - CLASIFICACIÓN DE ECG Y EXTRACCIÓN DE CARACTERÍSTICAS CON CONOCIMIENTO DE DOMINIO

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.

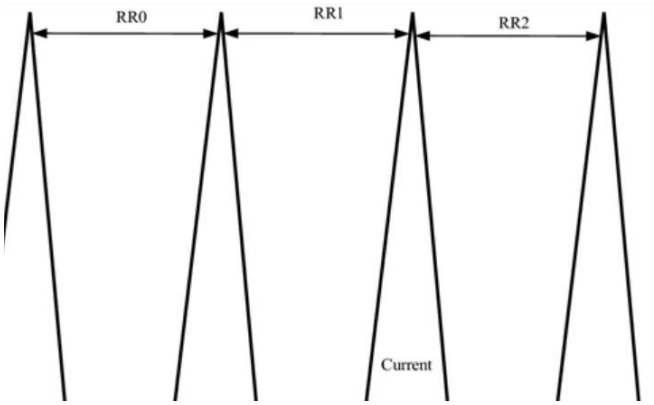
Por ejemplo, una señal con 18500 muestras se convierte en dos señales de 9000 muestras y las 500 muestras restantes se ignoran.

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.

Se pueden extraer o generar diferentes características a partir de los datos sin procesar. En este proyecto usamos las siguientes características:



$$Ratio1 = \frac{RR_0}{RR_1} \quad Ratio2 = \frac{RR_2}{RR_1} \quad Ratio3 = \frac{RR_m}{RR_1}$$

where

$$RR_m = mean(RR_0, RR_1, RR_2)$$

Agregamos 2 características más: la amplitud de los picos R a partir de los dos canales de señal en la base de datos, lo que nos lleva a un total de 8 características.

Cargar Datos de Características

Tarea: Cargue el archivo `ECG_ML_Features_DK.mat`. Cree una variable `variableNames` que contenga todos los nombres de las características. La última columna representa las etiquetas (abnorm = 0 significa que el latido del corazón es normal).

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.

En este proyecto, usaremos el 80% de los datos para el entrenamiento y el resto para las pruebas.

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.

Entrenar usando Classification Learner

Utilice la aplicación Classification Learner para entrenar, comparar y seleccionar clasificadores de forma interactiva. 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.

Una vez que se entrena el mejor modelo, puede exportarlo en el espacio de trabajo de MATLAB como variable `trainedModel`, o generar automáticamente la función para entrenar modelos de manera programática. Luego, puede usar el modelo exportado para hacer predicciones sobre nuevos datos (conjunto de prueba).

Para abrir la aplicación Classification Learner, puede ejecutar el comando

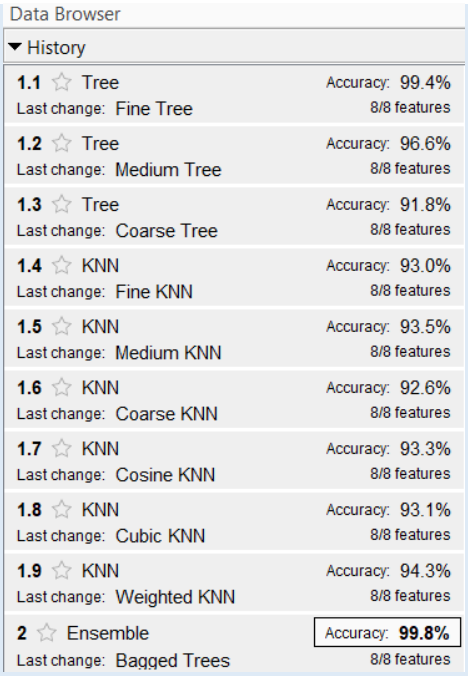
```
classificationLearner
```

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 `abnormal` esté seleccionada como `Response` y que todas las casillas de `Predictors` estén marcadas (excepto `abnormal`). 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 durar varios segundos.
- En la ventana `History`, 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`.

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

En general, los modelos funcionan muy bien con datos de prueba.

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

Resumen

En este proyecto, creamos modelos ML que identifican la arritmia cardíaca a partir de formas de onda de ECG. Al aprovechar el conocimiento del dominio médico y el procesamiento de señales, extrajimos características más poderosas. Entrenamos interactivamente algunos modelos con

Classification Learner y comparamos su desempeño. En general, el rendimiento de la mayoría de los modelos fue muy bueno (entre el 90 y el 99%).

Archivos requeridos:

`ECG_ML_Features_DK.mat`

`PhysionetData.mat`

`plotSignalLengths.m`

`segmentSignals.m`

`visualizeECGSignals.m`