Predicción de dolencias cardiacas a partir de un electrocardiograma

Machine Learning. PEC 4

Introducción

Se dispone del resultado de 1200 electrocardiogramas (ECG) en pacientes con algún de estos 4 tipos de problemas cardíacos: Arrhythmia (ARR), Congestive Heart Failure (CHF), Atrial Fibrillation (AFF), Normal Sinus Rhythm (NSR). El número de variables recogidas es de 54. La primera variable `RECORD` se usa como identificador. La ultima variable corresponde a la clase y el resto son las variables explicativas de tipo numérico. Como muestra se presentan las 8 primeras:

- hbpermin: Heart beat per minute
- Pseg: P wave segment length in ms
- PQseg: PQ segment length in ms
- QRSseg: QRS segment length in ms
- QRseg: QR segment length in ms
- QTseg: QT segment length in ms
- RSseg: RS segment length in ms
- STseg: ST segment length in ms

2. Objetivo

El objetivo de este análisis es **predecir el tipo de dolencia cardíaca de los pacientes** a partir de la información recogida en el ECG. Los datos se encuentran en el fichero ECGCvdata.csv. La columna 56 contiene la clase.

En esta PEC se analizan los datos mediante la **implementación** de los diferentes **algoritmos estudiados**: *k-Nearest Neighbour, Naive Bayes, Artificial Neural Network, Support Vector Machine, Arbol de Decisión y Random Forest* para **predecir** el tipo de docencia cardíaca.

Puntos importantes:

1. La PEC se puede implementar en **R** o en **Python** o, **combinando ambos**. El informe de la PEC se realizará en Rmarkdown o notebook para Python.
2. Se debe aplicar la misma selección de datos training y test en todos los algoritmos. Utilizando la semilla aleatoria `12345`, para separar los datos en dos partes, una parte para training (67%) y otra parte para test (33%). Si se prefiere, se puede escoger otro tipo de partición de los datos para hacer la selección de training y test como por ejemplo k-fold crossvalidation, boostrap, random splitting, etc. Lo que es importante es mantener la misma selección para todos los algoritmos.
3. Realizar una exploración de los datos que incluya una estadística descriptiva básica de las variables mediante tablas y gráficos. Si hay valores ausentes ("missing") se eliminaran las variables que los contengan.
4. En todos los casos se evalúa la calidad del algoritmo con la información obtenida de la función `confusionMatrix()` del paquete `caret` o equivalente en Python.
5. Para la ejecución especifica de cada algoritmo se puede usar la función de cada algoritmo como se presenta en el libro de referencia o usar el paquete `caret` con los diferentes modelos de los algoritmos.

O incluso, hacer una versión mixta.

6. Comentario sobre el informe dinámico en **R**. Una opción interesante del knitr es poner `cache=TRUE`. Por ejemplo:

   ```
   knitr::opts_chunk$set(echo = FALSE, comment = NULL, cache = TRUE)
   ```

   Con esta opción al ejecutar el informe dinámico crea unas carpetas donde se guardan los resultados de los procesos. Cuando se vuelve a ejecutar de nuevo el informe dinámico solo ejecuta código R donde se ha producido cambios, en el resto lee la información previamente descargada. Es una opción muy adecuada cuando la ejecución es muy costosa computacionalmente.

Informe de la PEC

Las soluciones se presentarán mediante un informe dinámico Rmarkdown o notebook con la siguiente estructura:

1. Titulo: igual que el de la PEC, autor, fecha de creación e indice de apartados de la PEC.
2. Sección de lectura, exploración de los datos y obtención de los muestras de train y test. Recordar que un primer paso es, si hace falta, transformar las variables leídas al tipo de objeto R y/o Python adecuado al tipo de variable. La exploración de los datos se aplica a todas las variables leídas. Es adecuado realizar gráficos univariantes o multivariantes para realizar este apartado. (*Puntuación*: 10%)
3. Sección de aplicación de cada algoritmo para la clasificación. Está formado por subsecciones que corresponden a cada algoritmo y en este orden:
   - k-Nearest Neighbour. Se explorarán los valores para el número de vecinos $k = 1, 3, 5, 7, 11$.
   - Naive Bayes. Se explorará la opción de activar o no `laplace`.
   - Artificial Neural Network. Se explorarán las arquitecturas con una y dos capas ocultas: 1) con 15 nodos en una capa oculta, 2) 25 y 10 nodos en cada capa oculta.
   - Support Vector Machine. Se explorarán la funciones kernel lineal y rbf.
   - Árbol de Clasificación. Se explorará la opción de activar o no `boosting`.
   - Random Forest. Se explorará la opción de número de árboles $n = 100, 200$.

   La implementación puede realizarse en R y/o Python. (*Puntuación*: 60%)

   En cada algoritmo hay que realizar las etapas mencionadas anteriormente y presentar una tabla de rendimiento de la calidad de la clasificación con al menos tres métricas para los opciones exploradas y un breve comentario.
4. Sección de conclusión y discusión sobre el rendimiento de los algoritmos con al menos tres métricas para el problema tratado. Proponer que modelo o modelos son los mejores. (*Puntuación*: 20%)

Una característica que se valorará es hasta que punto es el informe "dinámico". En el sentido de adaptarse el informe a cambios en los datos, es decir, si el fichero de datos cambia el informe se adapta a los nuevos resultados. (*Puntuación*: 10%)

Se subirán al registro de entregas un **zip** con los siguientes ficheros:

1. Fichero ejecutable (.Rmd o .ipynb) que incluya un texto explicativo que detalle los pasos implementados en el script y el código de los análisis. No olvidar de incluir todos los ficheros complementarios que hagan falta para la correcta ejecución: *ficheros de datos, fichero de bibliografia, imágenes, ...*

   NOTA: Para facilitar la ejecución, no usar un ruta fija para la lectura del fichero, asociarlo al area de trabajo donde este el fichero .Rmd o .ipynb
2. Informe (pdf / html) resultado de la ejecución del fichero Rmd o .ipynb anterior.

Antes de enviar el zip, se recomienda **verificar la reproducibilidad del fichero .Rmd o .ipynb** para obtener el informe en formato pdf sin ninguna dificultad.