Trabajo sobre Clasificación en el paquete `caret`

José Tomás Palma Méndez

Dept. of Information and Communication Engineering. University of Murcia
Contacting author: `jtpalma@um.es`

Para este trabjo vamos a partir del fichero echo.kNNImpute.csv generado en el trabajo de evaluación anterior.

Ejercicio 1. Genera un gráfico de dispersión para las variables disponibles en el fichero. Prueba todas las opciones del parámetro pairs, box, strip, density y ellipse.

Ejercicio 2.

2.a) ¿Cómo podríamos generar un conjunto de datos qué solo contenga las componentes principales seleccionadas y el atributo que identifica la clase?
2.b) Crea un gráfico en el qué se muestren las cuatro primeras componentes principales del conjunto de entrenamiento.
2.c) Repite el proceso seguido para el ACP para determinar las componentes independientes.
2.d) Crea un gráfico en el qué se muestren las cuatro primeras componentes principales del conjunto de entrenamiento.
2.e) Al realizar el análisis ACP o ICA ¿Qué otros métodos de preprocesamiento se han aplicado?
2.f) ¿Existe alguna variable que presente una varianza cercana a 0?

Ejercicio 3. Realiza una selección de variables utilizando la función `rfe` con las funciones `treebagFuncs` y `nbFuncs` (Recuerda que a lo mejor es necesario volver a reprocesar el conjunto de datos).

3.a) ¿Cuántas variables se seleccionan con cada técnica?
3.b) Compara estos resultados aplicando la misma técnica pero con las modelos de clasificación `svmLinear` y `rpart` (recuerda que esto se debe hacer a través de la opción `functions=caretFuncs` y el parámetro `method` de la función `rfe`.
3.c) Se valorará, siempre de que se dispongan de recursos de computación adecuada, la aplicación de otros métodos de selección de variables.
3.d) Aplica la selección por filtros utilizando las funciones de evaluación disponibles e indica cuantas variables se seleccionan en cada caso.
3.e) Basándote tu criterio, selecciona el mejor conjunto de variables generado por los métodos anteriores, genera un conjunto de prueba y otro de entrenamiento.

Ejercicio 4. Utiliza el conjunto de entrenamiento para generar los siguientes clasificadores: `rpart`, `svmLinear`, `knn`, `lda` y `nnet`.

4.a) Los nombres de cada uno de los modelos deben ser `rpartFit`, `svmFit`, `knnFit`, `ldaFit` y `nnetFit` respectivamente.
   - En los casos en los que sea posible prueba además diferentes combinaciones de técnicas de preprocesamiento.
   - Define diferentes conjuntos de combinación de parámetros para los métodos que lo permitan y genera el modelo con la mejor combinación.

4.b) Crea una tabla en que contenga los resultados de la precisión y el índice Kappa para cada uno de los modelos. Las filas llevarán el nombre de cada uno de los modelos y las columnas el nombre del índice correspondiente.

4.c) Repite los pasos anteriores pero ahora realiza la evaluación utilizando como métrica el Área Bajo la Curva ROC.

4.d) Cálculo la curva ROC para el modelo que mejor rendimiento ofrezca.

4.e) Calcula las predicciones y las probabilidades en la asignación de clases para el modelo que presente mejor rendimiento.

Ejercicio 5. Elige los tres modelos más prometedores de los anteriormente generados (ambos tienen que haber sido evaluados con la misma técnica).

5.a) Calcula la matriz de confusión y los principales índices de eficiencia de ambos modelos y genera una tabla con los nombres de los modelos en las filas y el de los indicadores en las columnas.

5.b) Muestra los resultados de forma gráfica.

5.c) Compara los dos modelos con mejor precisión y selecciona el mejor.

5.d) Compara todos los modelos y selecciona el mejor.

Ejercicio 6. Utilizando el mismo conjunto de datos que en el trabajo sobre clasificación, construye un clasificador para cada uno de las técnicas descritas anteriormente: `C5.0` (para la versión de reglas), `rfRules`, `JRip` y `PART`, probando con diferente configuración de parámetros.

6.a) Genera una tabla en la que aparezca la precisión y el índice Kappa (junto sus desviaciones típicas), asegurándote de seleccionar la mejor configuración.

6.b) Realiza la misma operación pero utilizando como medidas el índice ROC, la especificidad y la sensibilidad (junto a sus desviaciones típicas).

6.c) Compara los modelos y selecciona el mejor.

6.d) Compara el mejor modelo con el mejor obtenido en el trabajo sobre el paquete `caret`.

Técnicas basadas en Reglas en R

En `R` tenemos algunos paquetes que pueden ser llamados desde `caret` y que nos permiten obtener clasificadores o regresores basados en reglas:

- `C5.0`. Este paquete implementa el algoritmo para construir árboles de decisión `C5.0`. Tienes varias funciones que nos permiten ejecutar el algoritmo de diferentes maneras:
  - `C5.0`: es la implementación clásica y tiene los siguientes parámetros: `trials`, que es el número iteraciones al aplicar *boosting*, `model`, que permite indicar si se lo que se quiere obtener un árbol (`"tree"`) o un conjunto de reglas (`rules"`) y `winnow`, para indicar se se incluye el proceso de selección de variables antes de realizar cada partición[1].
  - `C5.0Cost`: tiene los mismos parámetros que el método anterior y añade el parámetro `cost` que permite indicar el coste de cada uno de los tipos de error mediante una matriz de dimensión CxC sinedo C el número de clases.
  - `C5.0Rules`, genera un conjunto de reglas sin utilizar *boosting* ni selección de variables interna, *winnow*.
- **Random Forest Rule-Based Model** (`rfRules`). Mediante esta técnica podemos extraer un conjunto de reglas a partir de un conjunto de árboles generados mediante la técnica Random Forest.Los parámetros que podemos utilizar para ajustar el modelo son: `mtry`, número de variables seleccionadas aleatoriamente para determinar la partición, y `maxdpeth`, profundidad máxima del conjunto de reglas. Este modelo también se puede utilizar para problemas de regresión.
- `JRip`. Este método está accesible a través del paquete `Rweka` y es una versión optimizada del algoritmo RIPPER. El único parámetro que tiene el `NumOpt` y que indica el número de veces que se intentan optimizar cada regla.
- `PART`. Esta técnica también pertenece al paquete `Rweka` e implmenta el algortimo PART y tiene dos parámetros `threshold` que indica el umbral de confidencia utilizado en el proceso de construcción de los árboles parciales, y `pruned` ("`yes`" o "`no`") para determinar si se aplica el proceso de poda.

Otros métodos basados en reglas que se ofrecen son `OneR` que es el clasificador por defecto que asigna siempre la clase más probable. Se pude utilizar como método base para compararlo con otros modelos.

También podemos encontrar modelos basados en reglas para problemas de regresión: `cubist`, `M5` y `M5Rules`, además de otras muchas técnicas basadas en lógica borrosa.

[1] Estos parámetros se pueden fijar al definir un grid mediante la función `expand.grid()`