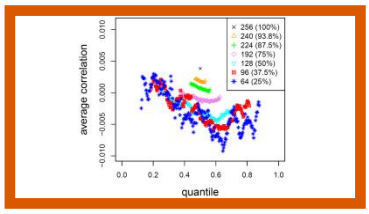
Asociación Española para la Inteligencia Artificial (**AEPIA**)

UIMP Universidad Internacional Menéndez Pelayo

Máster Universitario en Investigación en Inteligencia Artificial



CIENCIA DE DATOS Y APRENDIZAJE AUTOMÁTICO

Práctica 2: Evaluación de modelos

José Hernández-Orallo, DSIC, UPV, jorallo@dsic.upv.es

Basado en material de M.José Ramírez Quintana's

Vamos a trabajar con el conjunto de datos tic-tac-toe del repositorio UCI (y disponible en la plataforma). Esta base de datos contiene el conjunto completo de configuraciones del tablero al final de partidas de tres en raya, donde se asume que "x" es quien mueve primero. El concepto a aprender es "win for x" (es decir, cierto cuando "x" tiene uno de las 8 posibles maneras de crear tres en raya. El conjunto de datos consiste en 958 instancias (65.3% son positivas) y 9 atributos (x= ha jugado "x", o= ha jugado "o" (el segundo jugador), b= blanco):

1. top-left-square: {x,o,b}
2. top-middle-square: {x,o,b}
3. top-right-square: {x,o,b}
4. middle-left-square: {x,o,b}
5. middle-middle-square: {x,o,b}
6. middle-right-square: {x,o,b}
7. bottom-left-square: {x,o,b}
8. bottom-middle-square: {x,o,b}
9. bottom-right-square: {x,o,b}
10. Class: positive,negative

Para crear los modelos vamos a usar el paquete "caret" (http://topepo.github.io/caret/index.html). Este paquete proporciona herramientas útiles para la partición de datos, la generación de modelos y su evaluación, entre otras cosas. Vamos a construir los modelos usando validación cruzada (con 10 pliegues y 1 repetición) y generaremos 5 modelos diferentes de clasificación: Naïve Bayes, Decision Tree, aNeural Network, k-Nearest Neighbours y Support Vector Machine (con un *kernel* lineal), usando los datos de entrenamiento. Finalmente, evaluaremos los modelos usando el conjunto de test.

1. Carga los datos en R. Nombra las columnas para identificar major el tablero, como si se visitarán de izquierda a derecha y de arriba a abajo. Comprueba si hay valores faltantes.

2. Lee la sección "data splitting" de la web de "caret". A continuación parte los datos en 70% para entrenamiento y 30% de test manteniendo la proporción original de clases.

3. Lee la sección "*the model training and parameter tuning*" de la página web de "caret" para saber cómo entrenar un modelo usando validación cruzada. Lee también la lista de modelos disponible en el enlace "Sortable Model List" con el fin de obtener el número del método que implementa cada técnica, así como el paquete que debemos instalar para usarla. Utiliza la misma semilla para entrenar los modelos para hacer una comparación más justa de su desempeño. No modifiques los parámetros de ajuste (tuning) por defecto de la función *train*. Los valores de las medidas de evaluación para seleccionar el mejor modelo se muestran con simplemente escribir el nombre del modelo en la consola del R. Por defecto, el porcentaje de aciertos (*accuracy*) y/o la medida kappa se utilizan para modelos de clasificación). Una manera alternativa de recoger los resultados de remuestreo es mediante la función *resamples* (también del paquete "caret"). Completa la tabla siguiente con los valores finales de "accuracy" y "kappa" para los datos de entrenamiento:

	Accuracy	Kappa
Naive Bayes		
Decision Tree		
Neural Network		
Nearest Neighbour		
SVM (linear kernel)		

4. Lee la sección "model performance" de la página web de "caret". Aplica los modelos al conjunto de datos de test. Muestra la matriz de confusión y observa la información que proporciona. Construye una tabla (similar a la anterior) con los valores de accuracy y kappa obtenidos para cada modelo en el conjunto de test (para esto puedes usar la función *postResample*). Añada a la tabla el valor AUC. Se puede calcular usando el paquete *AUC* (https://cran.r-project.org/web/packages/AUC/AUC.pdf).

5. Representa las curvas ROC para los modelos. Usaremos el paquete *ROCR* (http://rocr.bioinf.mpi-sb.mpg.de/ROCR.pdf):
    a. Primero, calculamos de Nuevo las predicciones del conjunto de test pero ahora cambiamos el parámetro de la función *predict* a "prob" con el fin de generar un clasificador probabilístico (un estimador de probabilidades).

b. Construimos un objeto "*prediction*" para cada clasificadores usando el vector de probabilidades para la clase positive como primer parámetros, y el vector de la clase real como segundo parámetro.
c. Calculamos las medidas que queremos mostrar en una gráfica en el eje y (TPR) y en el eje x (FPR) mediante la función *performance*.
d. Dibujamos todas las curvas en la misma gráfica.

Las curvas deberían ser como sigue: