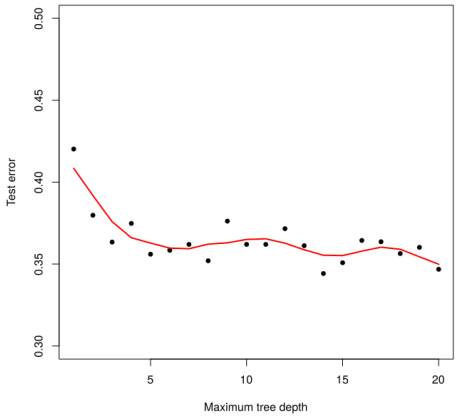
# Trabajo Práctico 4
# Métodos Supervizados Avanzados

AGUSTÍN MISTA
*Universidad Nacional de Rosario*
*Tópicos de Minería de Datos*
Rosario, 3 de Diciembre de 2017

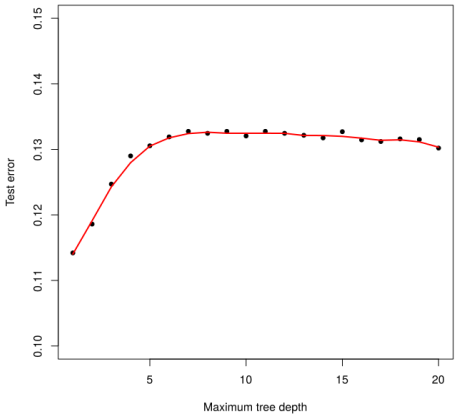
## Ejercicio 1.

Para este primer ejercicio pusimos a prueba la performance del método de clasificación basado en boosting disponible en la librería `adabag` de R. Para ésto utilizamos datasets correspondientes al problema de las *espirales anidadas* y el problema *diagonal*, contando en en ambos casos con conjuntos separados de entrenamiento y test. En cada caso, evaluamos como la complejidad de los árboles presentes en cada ensemble (en torno a la máxima profundidad de los mismos) afecta el error de test sobre nuestros datasets.

A continuación se muestran los resultados obtenidos sobre ambos datasets, utilizando 200 árboles para cada ensemble, y variando la profundidad máxima de los mismos entre 1 y 20 niveles.




Espirales anidadas




Diagonal

En el caso del problema de las espirales anidadas, puede observarse que los resultados son bastante pobres. Respecto de la profundidad máxima de los árboles, el error tiende a decrecer para valores entre 1 y 5, para luego estancarse (o decrecer muy lentamente). Este comportamiento podría deberse en un principio a que este dataset requiere de un cierto número mínimo de reglas de decisión para lograr que el clasificador pueda efectivamente discernir en qué espiral se encuentra cada punto a medida que las mismas se entrelazan formando bucles.

Por otro lado, para el caso del problema diagonal los resultados son considerablemente mejores, debido posiblemente a que este dataset es ciertamente más sencillo de clasificar que el anterior. La curva de error respecto de la profundidad máxima de los árboles crece levemente cuando se aumenta entre 1 y 5 la profundidad máxima, para luego estabilizarse.

En ambos casos, aumentar desmedidamente la complejidad de los árboles en los ensembles no parece favorecer los resultados finales respecto del error de clasificación del conjunto de test.

## Ejercicio 2.

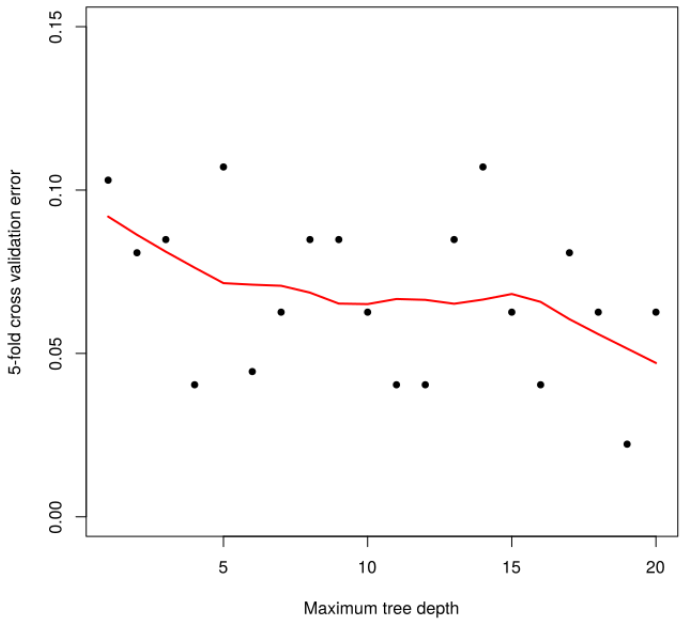
En este segundo ejercicio ponemos a prueba la capacidad de clasificación de los métodos Random Forest, Boosting y Support Vector Machines ante el dataset Lampone. Una vez eliminado el año de medicion, junto a aquellas columnas nulas, constantes o no numéricas, este dataset posee 49 muestras y 127 features, de las cuales nos interesa predecir la clase de cada muestra dada por la feature binaria `N_tipo`.

Puesto que este dataset posee un número muy acotado de muestras, y no se posee un conjunto de test apropiado, se decidió evaluar el error de test efectuando 5-fold cross validation y tomando la media del error sobre cada fold para cada uno de los métodos. A continuación se listan los resultados obtenidos para cada uno de ellos.

**Random Forest** Éste método fue ejecutado sin optimizar ningún parámetro de entrada. Con el mismo se obtuvo un error medio de clasificación de **0.084** sobre cada uno de los folds realizados. Éste resultado parece ser aceptable dada la pequeña cantidad de muestras disponibles para entrenar, y se supone que con un número de muestras mucho mayor, los resultados deberían mejorar considerablemente.

**Boosting** Para este método se utilizaron 100 árboles en cada ensemble y, al igual que en el ejercicio anterior, se optimizó la máxima profundidad de los mismos entre 1 y 20 niveles.

A continuación se muestra el error medio de clasificación respecto de la profundidad máxima de los árboles en cada ensemble, junto con una curva suavizada del mismo.

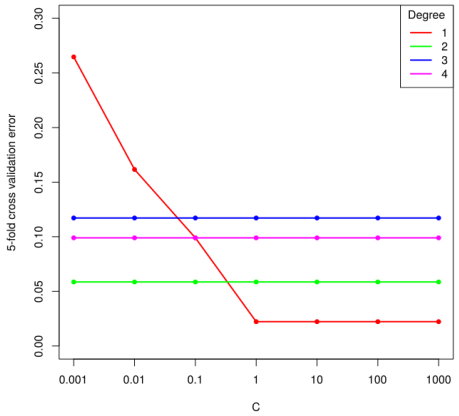



A primera vista, los resultados muestran un error de clasificación bastante ruidoso respecto de la profundidad máxima, lo que podría sugerir que hacen falta o bien más muestras en nuestro dataset, o bien incrementar la cantidad de árboles en cada ensemble para obtener resultados más robustos.

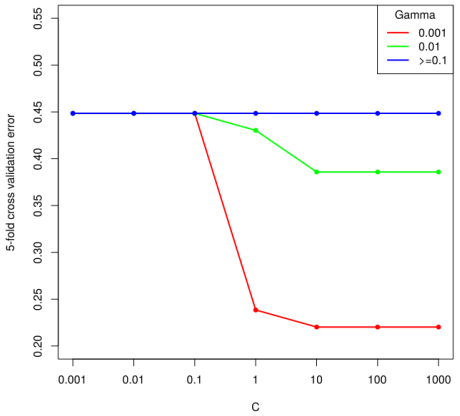
Por otro lado, si observamos la curva suavizada del error, podemos ver que los resultados muestran un error de clasificación levemente mejor al caso anterior ($\sim 6\%$) y que decrece lentamente a medida que se aumenta la profundidad máxima de los árboles en cada ensemble.

**Support Vector Machines** Para este método se usaron los kernels `polinomial` y `radial`, optimizando en ambos casos el costo de violación de restricciones C usando $1 \times 10^i$ con $-3 \leq i \leq 3$. A su vez, en el caso del kernel polinomial se optimizó el grado del polinomio usado entre 1 y 4, mientras que para el kernel radial se optimizó el parámetro $\gamma$ de la función radial usando $1 \times 10^i$ con $-3 \leq i \leq 3$.

A continuación se muestran los resultados de error medio de clasificación obtenidos en función del costo C para cada elección del kernel y sus respectivos parámetros.




SVM con kernel `polynomial`




SVM con kernel `radial`

Para el caso del kernel polinomial podemos ver como la variación del costo C no afecta el error de clasificación para aquellos casos donde se usó un polinomio de grado mayor a 1. No así para el caso lineal (grado igual a 1), donde el error se ve afectado en gran medida por el costo elegido, obteniendo resultados sospechosamente mejores que el resto de los métodos utilizados ($\sim$ 2%). En general, éste kernel funciona mejor para polinomios de grado bajo.

Si consideramos ahora el kernel radial, podemos ver que obtuvo resultados bastante pobres. La capacidad de clasificación parece ser bastante azaroza cuando se utilizan valores del parámetro $\gamma$ mayores a 0.1 (error cercano al 50%) independientemente del costo C elegido. Por otro lado, la capacidad de predicción aumenta cuando se utiliza una combinación parámetros donde $\gamma$ es pequeño y C es mayor a 10, sin llegar a considerarse del todo buena (error $\sim$ 22%).

**Nota** Cabe remarcar nuevamente que este dataset posee un número muy reducido de muestras, y que nuestros análisis sobre el mismo pueden estar bastante sujetos a cuán adecuados sean los folds creados automátiamente usando la librería `dismo`. Para conseguir resultados más robustos hace falta inevitablemente poder acceder a un número mayor de muestras.