# Redes Neuronales y Aprendizaje Profundo para Visión Artificial

**Cuatrimestre:** 2do de 2020

---

## PRÁCTICA 10: MÉTODOS BASADOS EN ÁRBOLES DE DECISIÓN

---

1. Este práctico es una adaptación a Python de "'Introduction to Statistical Learning with Applications in R" de Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani. Parte del práctico es buscar y encontrar las herramientas adecuadas en `scikit-learn` para hacer los puntos, es decir, no les estamos pidiendo implementar nada de cero si no más bien usar las herramientas que ya están disponibles. Esto es así para entrenar, para estimar errores, para usar técnicas de muestreo, para hacer plots, etc.

   El práctico se centra en implementar árboles de decisión (y variantes de ensemble) en la base de datos `Carseats` (que pueden descargar acá ).

   En primer lugar, a partir de la variable `Sales` creen una nueva variable binaria que se llame `High`, que sea `Yes` si `Sales` es mayor a 8 o `No` si es menor que 8. A partir de ahí tienen que usar árboles de decisión para estimar `High` con clasificación y `Sales` con regresión (importante: tomar la precaución de sacar una al estimar la otra, ya que si no tendrán errores artificialmente bajos).

   (a) Separar los datos en dos particiones, una para datos de entrenamiento/validación (o sea, de desarrollo) y otra para test.

   (b) Entrenar un árbol de decisión para clasificación de la variable `High`. Hacer un plot del árbol (usar `scikit-learn`) e interpretar los resultados.

   (c) Entrenar un árbol de decisión para regresión de la variable `Sales`. Hacer un plot del árbol (usar `scikit-learn`) e interpretar los resultados.

   (d) ¿Cuál es el error de test que obtienen en cada caso? Comparar con el error de entrenamiento y determinar si tienen *overfitting* o no.

   (e) Para el árbol de regresión, usar *cross-validation* para determinar el nivel óptimo de complejidad del árbol. Busquen cómo usar en `scikit-learn` la técnica de *pruning* para mejorar la tasa de error de test.

   (f) Para el caso de regresión, usar el abordaje tipo *bagging* para mejorar el error de test. Comparar con el abordaje de un único árbol de decisión. Buscar en `scikit-learn` cómo determinar el orden de importancia de los atributos.

   (g) Usar *random forests* para mejorar los resultados datos. Comparar el error de test con los abordajes anteriores. ¿Cambia el orden de la importancia de los atributos? Hacer un plot con el error de test en función del del hiperparámetro `max_features` que limita el número de atributos a incluir en cada split. Hacer otro plot equivalente en función de `max_depth`.

(h) Hacer la misma regresión usando `AdaBoost` y comparar errores de test con lo obtenido con *Random Forest* en el punto anterior.