Machine Learning aplicado a las Ciencias Sociales

Clase 7. Análisis supervisado. Ensamble Learning - Bagging / Random Forest

factor~data
EIDAES_UNSAM

Repasando…

- Los modelos de árboles de decisión son herramientas para la clasificación y regresión que implican dividir o segmentar el espacio predictivo en regiones
  - Basados en la “partición” de la información de los datos a partir de nodos
  - Modelos simples y fáciles de interpretar
  - Top-down: empiezan desde un nodo inicial en el árbol y luego va partiendo sucesivamente el espacio predictivo. Cada partición se indica con dos nuevas ramas bajo el árbol.
  - Greedy: en cada paso se busca la mejor partición en un punto particular, no se observa hacia adelante y elige una partición que va a ser más útil en el futuro.
  - No tiene muy buena capacidad predictiva, no considera todas las posibles particiones del espacio de atributos en J cajas.
  - Son modelos poco robustos: un cambio pequeño en los datos que tenemos puede cambiar el árbol final estimado

Ensamble learning

- Formas de “ensamblar” muchos árboles de decisión para mejorar su performance
- Combinamos varios modelos de base para ampliar el espacio de hipótesis posible para representar los datos
- En general, más precisos que los modelos base

- Dos familias de métodos de ensamble:
  - **Métodos de averaging**
  - **Métodos de boosting**

Métodos de averaging

- Basados en construir estimadores independientes y luego promediar las predicciones.

- Agregar este set de observaciones suele dar mejor que cualquier estimador base ya que **reduce la varianza** de cualquier método de aprendizaje estadístico.

- Ejemplos: Bagging, Random Forest y Extra Trees

Métodos de boosting

- Los estimadores de base se construyen secuencialmente y se trata de reducir el sesgo del estimador combinado.

- La premisa es combinar varios modelos “débiles” para construir un ensamblaje más poderoso.

- Ejemplos: XGBoost, AdaBoost (clases 4 y 5)

Espacio de hipótesis

- Aprendizaje supervisado: hacer predicciones de la verdadera función de clasificación f aprendiendo el clasificador h.

- Buscamos en un cierto espacio de hipótesis H la función más apropiada para describir la relación entre nuestras características y el objetivo.

- Puede haber varias razones por las cuales un clasificador base no pueda lograr mayor exactitud al tratar de aproximar la función de clasificación real.

Espacio de hipótesis

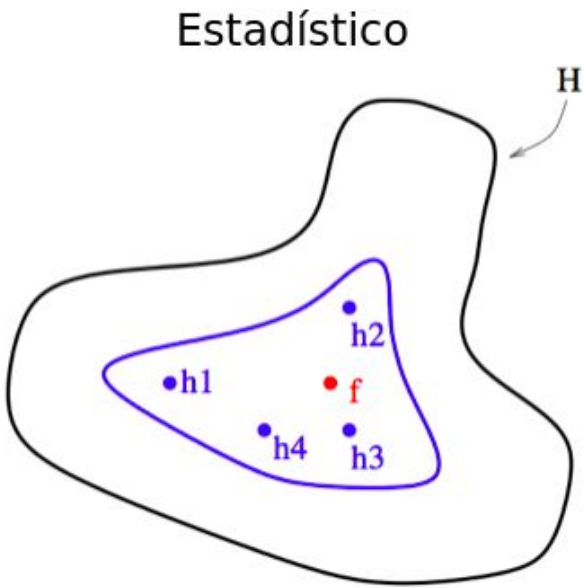
- Estos son tres de los posibles problemas:
    - Estadísticos
    - Computacionales
    - De representación

Problema estadístico

- Si la cantidad de datos de entrenamiento disponibles es pequeña, el clasificador base tendrá dificultades para converger a f .

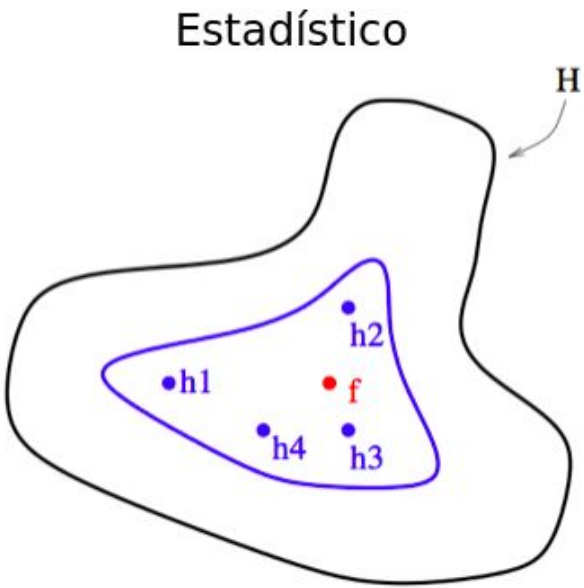
- Un ensamble puede mitigar este problema "promediando" las predicciones de los clasificadores.

- La función real f es mejor aproximada como un promedio de los clasificadores base hi.



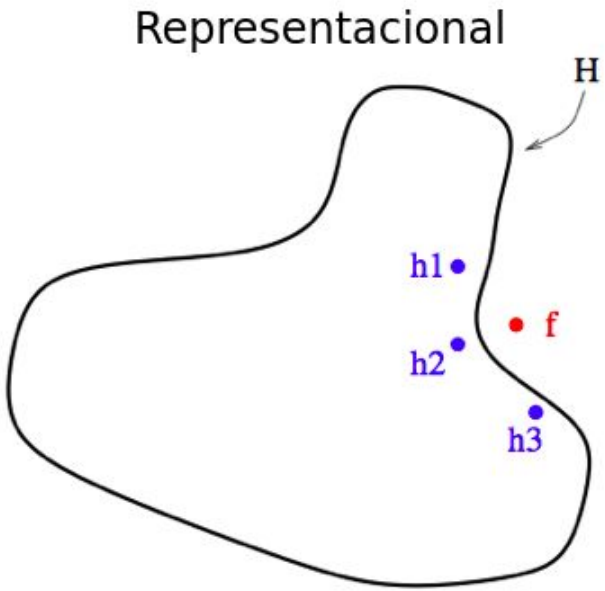
Problema computacional

- Puede ser computacionalmente difícil encontrar el mejor clasificador h.
- Imposible una búsqueda exhaustiva del espacio de hipótesis de todos los posibles clasificadores
- Un conjunto de varios clasificadores base con diferentes puntos de partida aproximar mejor f que cualquier clasificador base individualmente.



Problema de representación

- A veces f no se puede expresar en términos de la hipótesis.
- Si usamos un árbol de decisión como clasificador base, este trabaja formando particiones rectilíneas del espacio de características.
- Pero si f es una línea diagonal, entonces no puede ser representada por un número finito de segmentos rectilíneos. Por lo tanto, el límite de decisión verdadero, no puede ser expresado por un árbol de decisión.



¿En qué condiciones nos conviene usar los ensambles?

- **Capacidad predictiva:** los clasificadores base deben hacer mejores predicciones que la totalmente aleatoria. (Su AUC debe ser mayor a 0.5)
- **Diversidad:** los clasificadores base deben cometer distintos errores ante los mismos casos. (Sin diversidad no se puede mejorar la precisión del ensamble al combinar los clasificadores base)

Bagging

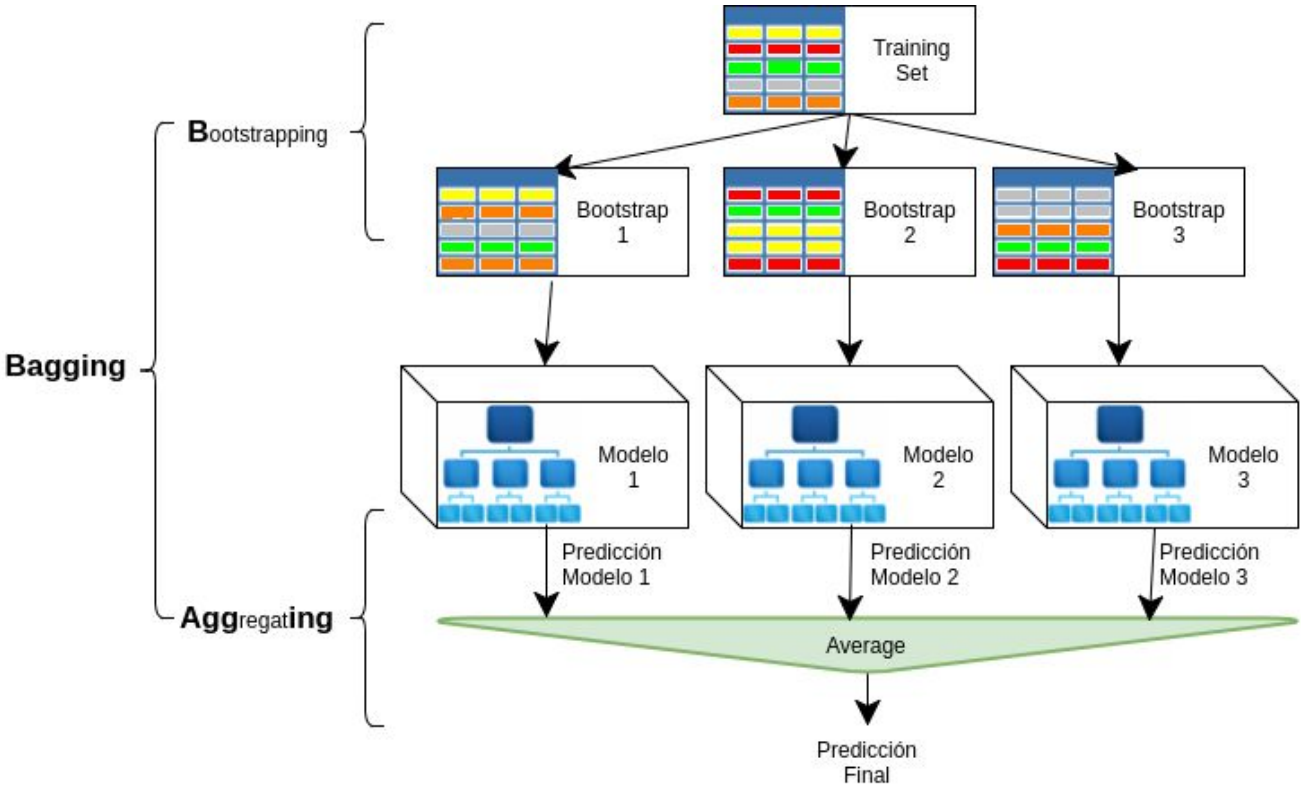
- Mencionamos que los árboles de decisión simples tienen alta variabilidad o, que es lo mismo, poca robustez → los resultados pueden variar mucho según la composición de los datos.
- Bagging (también llamado bootstrap aggregation) busca reducir esta varianza aleatorizando los datos en la construcción de varios modelos simples, para luego construir un árbol más fuerte promediando estos modelos simples.
- Promediar un set de observaciones reduce la varianza.

Bagging

- El modelo de bagging realiza *B* muestras de bootstrap del training set original,
- entrena clasificadores de base para cada una de esas sub-muestras (obtenemos *f^1(x), f^2(x), . . . , f^B(x)* funciones),
- devuelve las predicciones para cada modelo,
- promedia todas las predicciones para obtener la función final de predicción.

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{\star b}(x).$$

Bagging

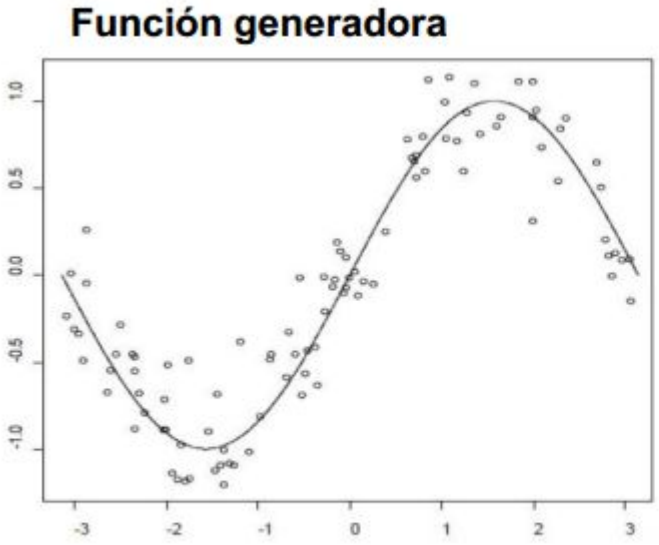

Bagging
Bootstrapping
Aggregating
Training Set
Bootstrap 1
Bootstrap 2
Bootstrap 3
Modelo 1
Modelo 2
Modelo 3
Predicción Modelo 1
Predicción Modelo 2
Predicción Modelo 3
Average
Predicción Final

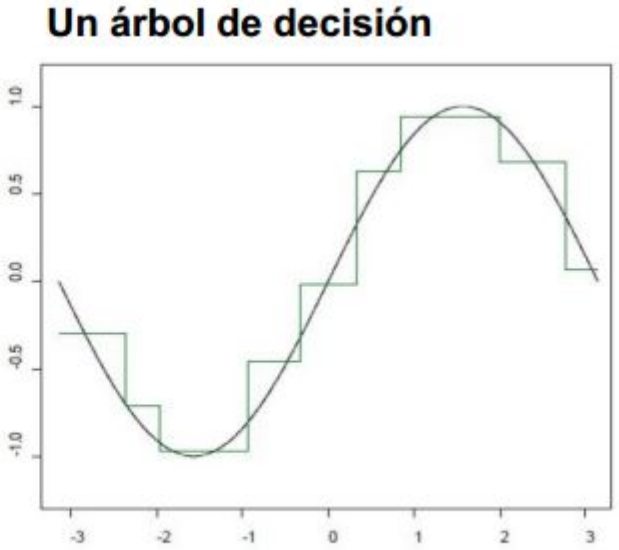
Bagging

Algorithm 5.6 Bagging Algorithm

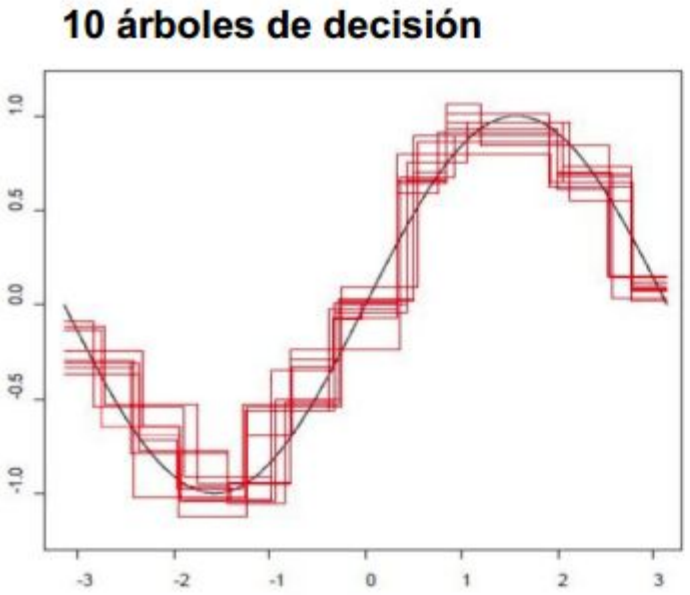
1: Let k be the number of bootstrap samples.
2: **for** $i = 1$ to k **do**
3: Create a bootstrap sample of size n, D_i.
4: Train a base classifier C_i on the bootstrap sample D_i.
5: **end for**
6: $C^*(x) = \arg\max_y \sum_i \delta\big(C_i(x) = y\big)$, $\{\delta(\cdot) = 1$ if its argument is true, and 0 otherwise.$\}$

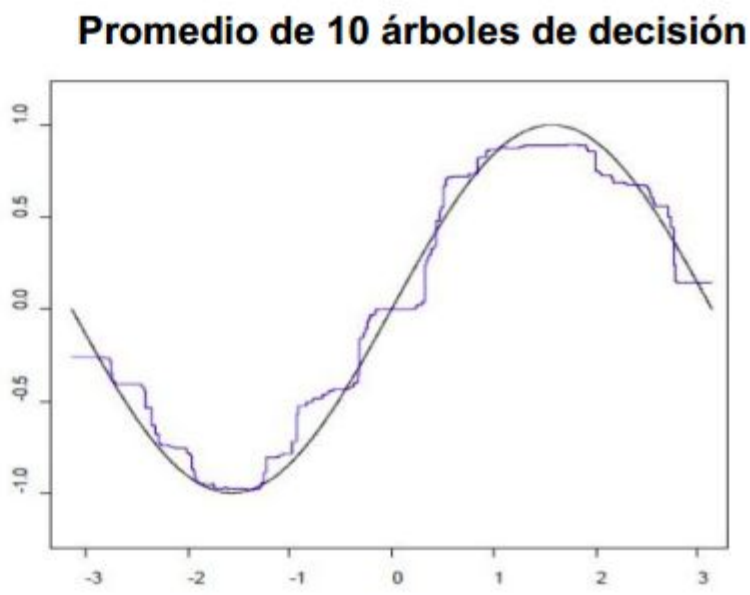
Bagging


Función generadora
1.0
0.5
0.0
-0.5
-1.0
-3
-2
-1
0
1
2
3


Un árbol de decisión
1.0
0.5
0.0
-0.5
-1.0
-3
-2
-1
0
1
2
3

Bagging


10 árboles de decisión
1.0
0.5
0.0
-0.5
-1.0
-3
-2
-1
0
1
2
3


Promedio de 10 árboles de decisión
1.0
0.5
0.0
-0.5
-1.0
-3
-2
-1
0
1
2
3

Bagging

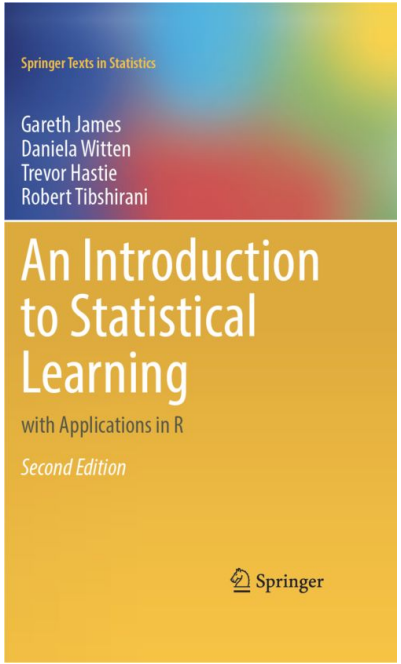
- El Bagging reduce la varianza del error de generalización al combinar múltiples clasificadores de base (siempre que estos satisfagan los requisitos anteriores).
- Ojo:
  - Si el clasificador base es estable, entonces el error del ensamble se debe principalmente al sesgo, y el bagging puede no ser efectivo.
  - Dado que cada muestra de datos de entrenamiento es igualmente probable, el bagging no es muy susceptible a overfitting con datos ruidosos.

Random Forest

- Consigna para resolver en clase: cada grupo (idealmente los mismos que van a presentar el trabajo final) va a armar dos materiales
- Una presentación breve (5 a 7 diapositivas) en las que explican el algoritmo Random Forest y las diferencias con Bagging
- Un código en el que se hace una implementación de Random Forest sobre el dataset de la ENUT

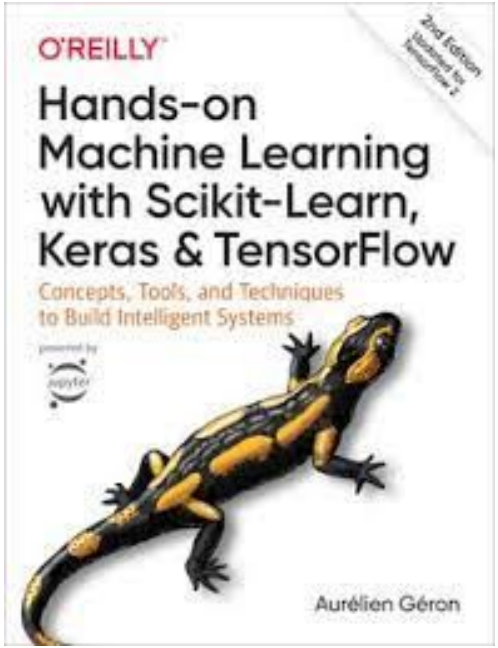
Random Forest

- Pueden usar cualquier material (libros, papers, posts, etc.)
- Algunas sugerencias:
  - Introduction to Statistical Learning Cap 8.2 (una especie de Aleph introductorio para Machine Learning)



Random Forest

- Pueden usar cualquier material (libros, papers, posts, etc.)
- Algunas sugerencias:
  - Hands on Machine Learning with Scikit-Learn Keras and Tensorflow. Concepts, Tools and Technique to Build Intelligent Systems. Cap. 7: el código está en Python pero la explicación teórica es sumamente amigable



Random Forest

- Pueden usar cualquier material (libros, papers, posts, etc.)
- Algunas sugerencias:
  - Tidy modeling with R. A framework for modeling in the Tidyverse, Cap. XX. Puede ser útil para la parte de implementación
  - Pueden revisar el canal de YouTube de Julia Silge con muchos tutoriales