

ÁRBOLES DE DECISIÓN II: *RANDOM FORESTS*

ALAN REYES-FIGUEROA

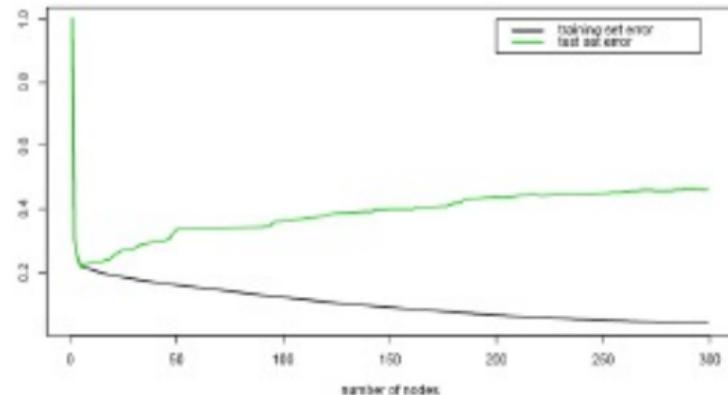
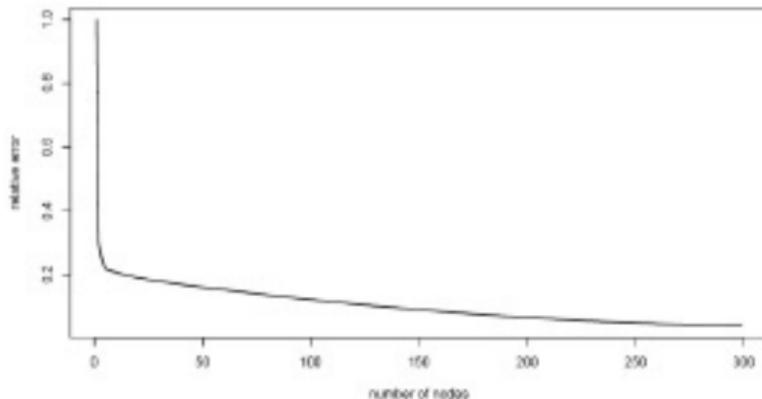
INTRODUCCIÓN A LA CIENCIA DE DATOS

(AULA 31) 06.MAYO.2021

Árboles de Decisión

Pregunta: ¿Dónde detenemos la construcción del árbol?

- Entre mayor profundidad, mejor ajuste a los datos (de entrenamiento)...
- ...pero no necesariamente mejor ajuste al conjunto de prueba.



Árboles de Decisión

Algunas ideas:

- Podemos incluir la complejidad del modelo en la función de costo.

$$C(T; \cdot) = \sum_i \beta_i I(R_i) + \alpha \text{size}(T),$$

donde

$$\text{size}(T) = \# \text{hojas}, \quad \text{size}(T) = \text{depth}(T).$$

- A mayor $\alpha > 0$, mayor penalización para árboles grandes.
- Podemos usar validación cruzada (próxima semana).

Propiedad

Si $T(\alpha)$ es un árbol óptimo para el valor α , entonces

$T(\alpha_1)$ es sub-árbol de $T(\alpha_2)$, si $\alpha_1 > \alpha_2$.

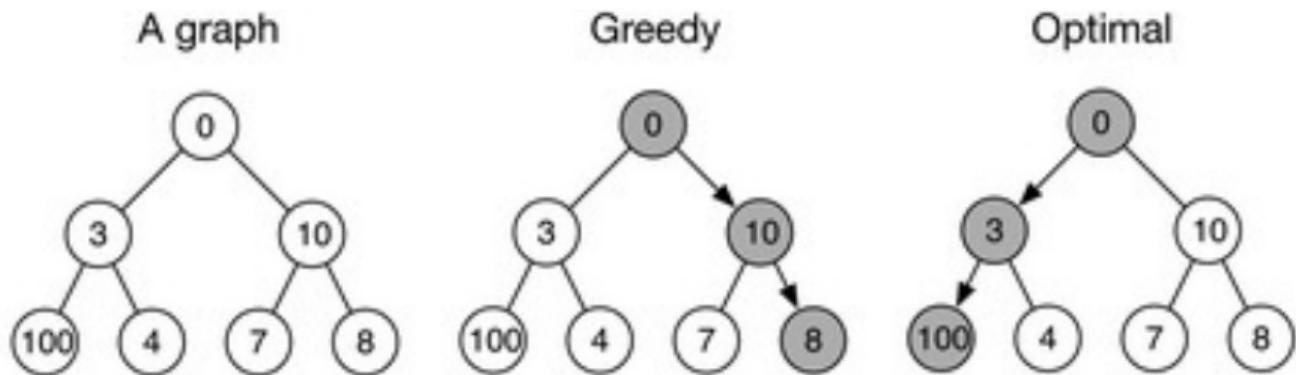
Lo anterior nos da un algoritmo eficiente para encontrar $T(\alpha)$ para cualquier $\alpha > 0$:

- Construir $T(0)$ (como antes).*
- Es suficiente limitarse a sub-árboles de $T(0)$ para encontrar $T(\alpha)$.*
- Elegimos $T(\alpha)$ en base de su poder predictivo aproximado con validación cruzada.*

Algoritmos Greedy

Propiedad:

- Los árboles de decisión son un ejemplo típico de *greedy optimization*.



A greedy algorithm fails to maximise the sum of nodes along a path from the top to the bottom because it lacks the foresight to choose suboptimal solutions in the current iteration that will allow for better solutions later

Importancia de variables

- Otra propiedad muy útil es que los árboles introducen un concepto de importancia de cada variable.

Definición

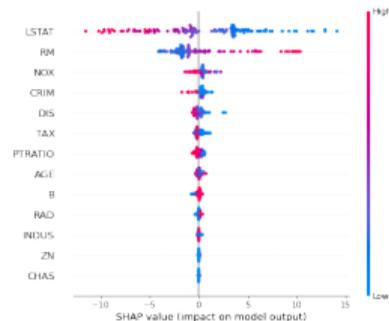
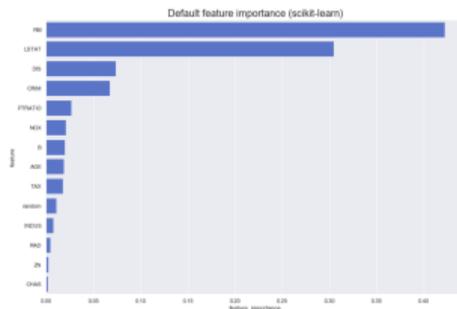
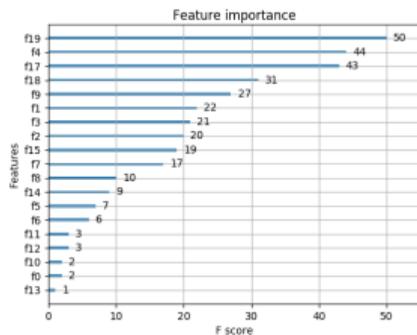
Sea T un árbol. Definimos la **importancia** ni de cada nodo $j \in T$ como la reducción de la impureza (ponderada) alcanzada en ese nodo

$$ni(j) = \frac{n_j}{n} (I(j) - \frac{n_{j,right}}{n_j} I(j_{right}) - \frac{n_{j,left}}{n_j} I(j_{left})),$$

donde n es el número total de muestras, n_j el número total de muestra que pasaron por el nodo j , y $n_{j,right}$, $n_{j,left}$ el número de muestras que pasaron por los hijo derecho e izquierdo de j , respectivamente.

La **importancia de la variable** x_k se calcula como la suma de importancias $\sum ni(j)$, sobre aquellos nodos donde la partición se hizo en la variable x_k .

Importancia de variables



Utilidad:

- Interpretación del modelo de decisión: cuáles variables están participando más y menos.
- De alguna manera, esto indica en qué se está fijando modelo (atención). Árboles son modelos de *caja blanca*.
- Reducción de la dimensionalidad.

Métodos de Ensamblaje

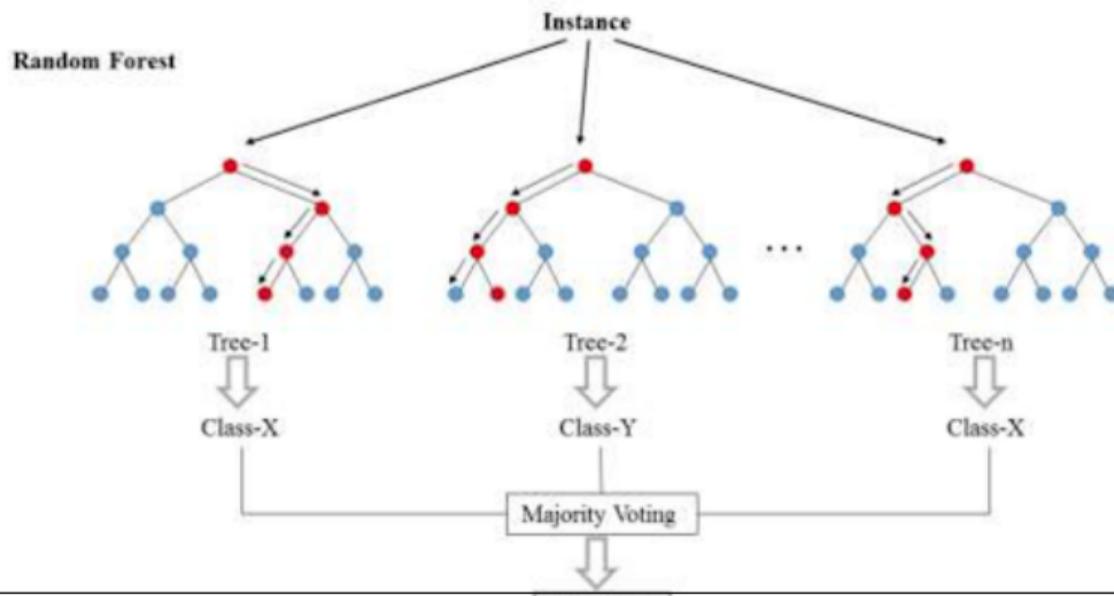
Los métodos de ensamblaje *ensemble learning* usan múltiples algoritmos de aprendizaje para obtener un mejor rendimiento predictivo que el que se podría obtener con cada método: combinar algoritmos o modelos simples para producir un mejor predictor.

Típicamente hay tres formas de combinar modelos:

- **Bagging:** Usa modelos homogéneos, de forma independiente unos de otros en paralelo y los combina siguiendo algún tipo de proceso determinista de promediado.
- **Boosting:** Usa modelos homogéneos, pero los aprende secuencialmente de forma muy adaptativa (un modelo base depende de los anteriores) y los combina siguiendo una estrategia determinista.
- **Stacking:** Usa modelos heterogéneos, los aprende en paralelo y los combina entrenando un metamodelo con esquemas de promediado.

Random Forests

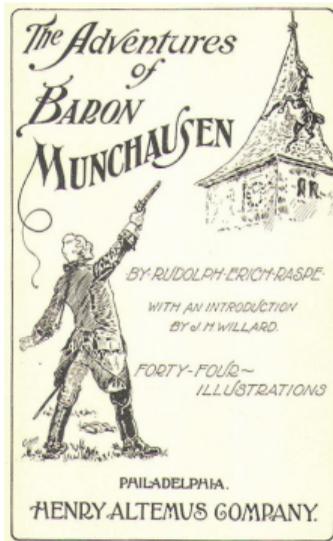
Los *Random forests* son modelos *ensemble*, en donde los modelos a combinar son árboles de decisión. Fueron introducidos por Leo Breimann (2001).



Bagging

Bagging = *Bootstrap aggregating*. En estadística, se refiere a utilizar una técnica de estimación por submuestreo llamada Bootstrap.

Rudolf Erich-Raspe *The Adventures of Baron Munchausen* (1781).



Bagging

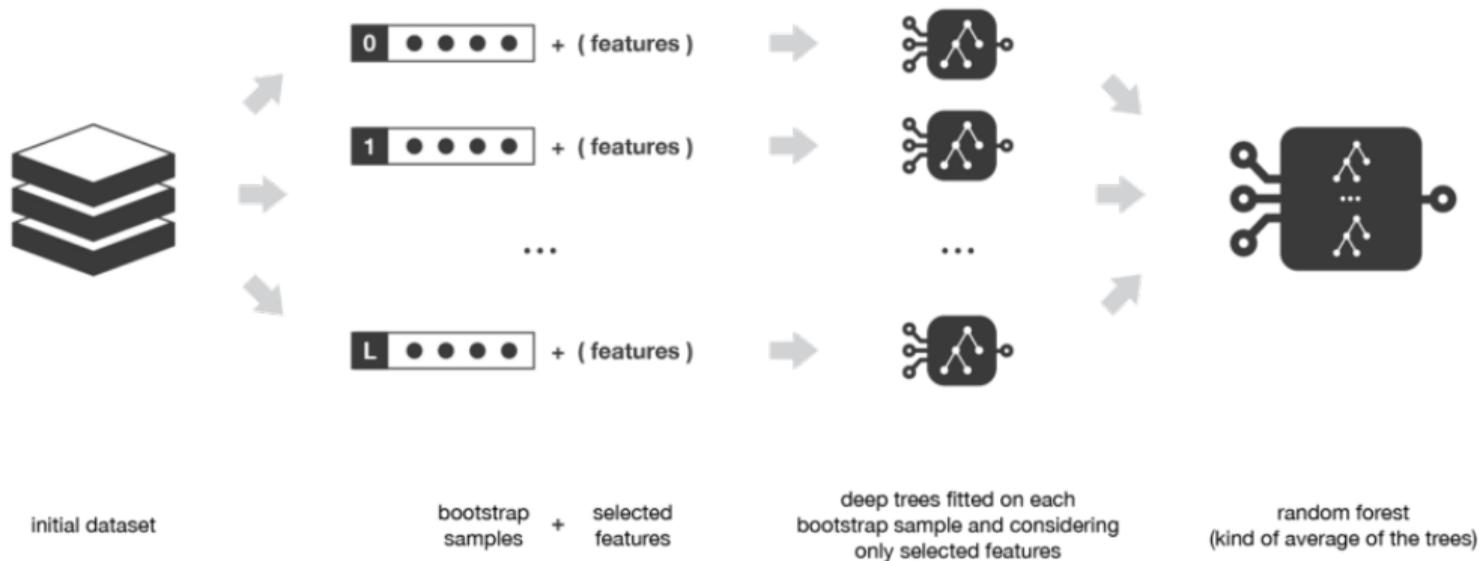


Ilustración gráfica de la técnica *bagging*.

Bagging

La ruta a seguir los modelos *random forest* es la siguiente:

1. Elegir un número N de modelos (árboles de decisión).
2. Por cada árbol $i = 1, 2, \dots, N$, hacer un submuestreo \mathbb{X}_i de los datos:
 - Submuestrear d_i variables del total d
 - Submuestrear n_i datos del total n de la muestra
 - Entrenar el modelo \hat{y}_i con la submuestra \mathbb{X}_i .
3. Combinar los N modelos \hat{y}_i en un metamodelo \hat{y} , mediante En clasificación:

$$\hat{y}(\mathbf{x}) = \operatorname{argmax}_j \left\{ \sum_{i=1}^N \mathbf{1}(\hat{y}_i(\mathbf{x}) = j) \right\}.$$

En regresión:

$$\hat{y}(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^N \hat{y}_i(\mathbf{x}).$$

Boosting

+  train a weak model
and aggregate it to
the ensemble model

 update the training dataset
(values or weights) based on the
current ensemble model results

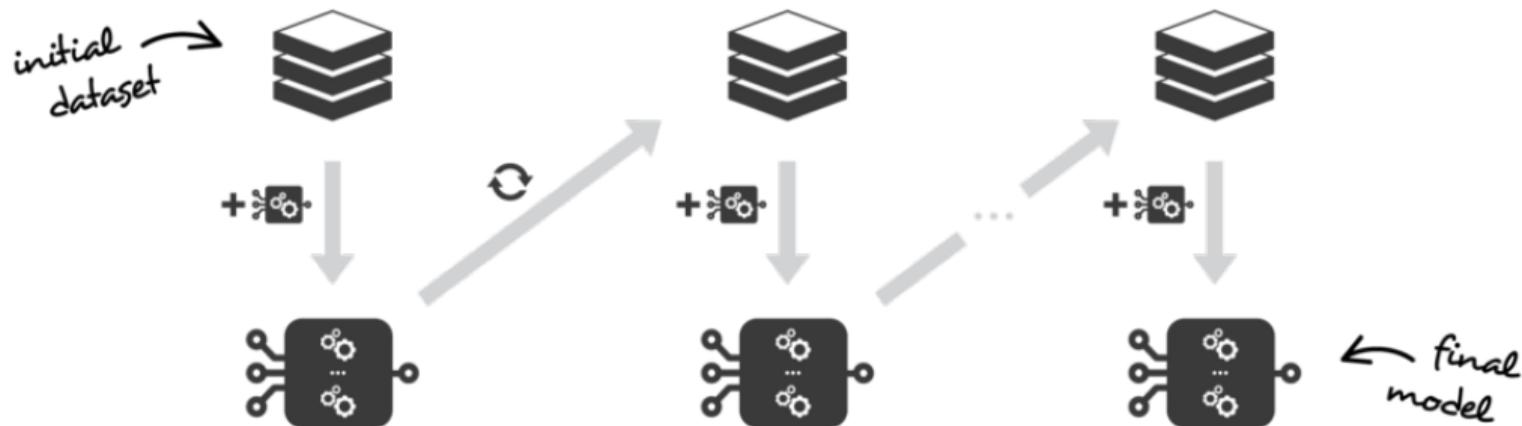


Ilustración gráfica de la técnica *boosting*.

Boosting



train a weak model and aggregate it to the ensemble model



update the weights of observations misclassified by the current ensemble model

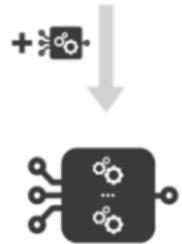
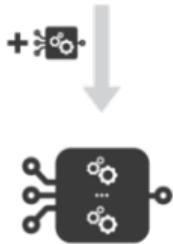
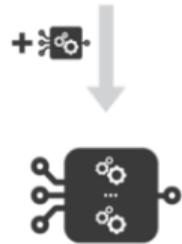
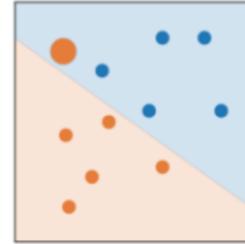
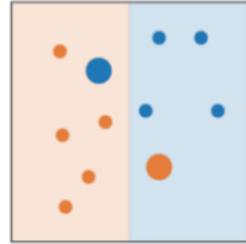
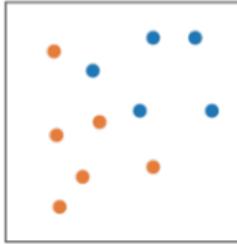


current ensemble model predicts "orange" class



current ensemble model predicts "blue" class

initial setting:
all the observations have the same weight



...

Boosting

Boosting funciona más como método de refuerzo: ajustan el modelo anterior, de forma adaptativa, en función de corregir los errores.

AdaBoost: (*Adaptive Boosting*): Definimos el modelo de (clasificación/regresión) como una suma ponderada de varios modelos simples

$$\hat{y}(\mathbf{x}) = \sum_{i=1}^N w_i \tilde{y}_i(\mathbf{x}).$$

Hallar el mejor modelo con esta forma es un problema de optimización difícil. En lugar de eso, se hace uso de un proceso de optimización iterativo mucho más tratable: en cada iteración, buscamos el mejor par posible (w_i, \hat{y}_i) para agregar al modelo actual.

$$\hat{y}_i(\mathbf{x}) = \hat{y}_{i-1}(\mathbf{x}) + w_i \tilde{y}_i(\mathbf{x}), \quad i = 1, 2, \dots, N.$$

Boosting

donde w_i y \tilde{y}_i se eligen de forma que \tilde{y}_i es el modelo que mejor ajusta los datos de entrenamiento, y así la mejor adaptación de \hat{y}_{i-1}

$$(w_i, \hat{y}_i) = \operatorname{argmin}_{w, \tilde{y}} \mathbb{E} L(\hat{y}_i + w \tilde{y}) = \operatorname{argmin}_{w, \tilde{y}} \sum_{j=1}^n L(\hat{y}_i(\mathbf{x}_j) + w \tilde{y}(\mathbf{x}_j)).$$

Gradient Boosting: Para la optimización, empleamos técnicas de descenso gradiente

$$\hat{y}_i(\mathbf{x}) = \hat{y}_{i-1}(\mathbf{x}) - \alpha \nabla_{\hat{y}_{i-1}} \mathbb{E} L(\hat{y}_{i-1})(\mathbf{x}).$$

Hoy en día existen librerías que ya hacen todo el trabajo:

- *scikit-learn*: AdaBoostClassifier, AdaBoostRegressor.
- XGBoost.

Boosting

-  train a weak model and aggregate it to the ensemble model
-  update the pseudo-residuals considering predictions of the current ensemble model
-  dataset values
-  predictions of the current ensemble model
-  pseudo-residuals (targets of the weak learner)



Stacking

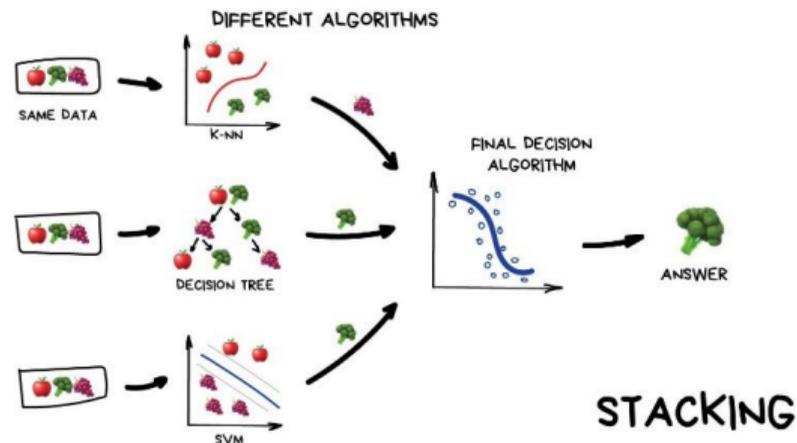
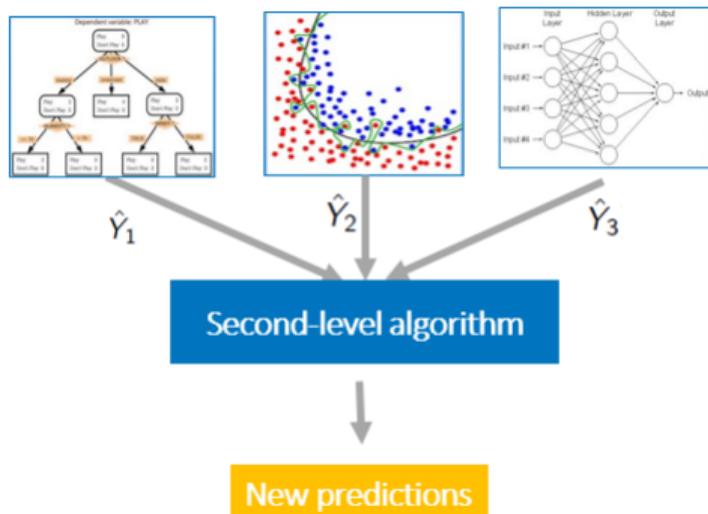


Ilustración gráfica de la técnica *stacking*.