Árboles de Decisión - Random Forest

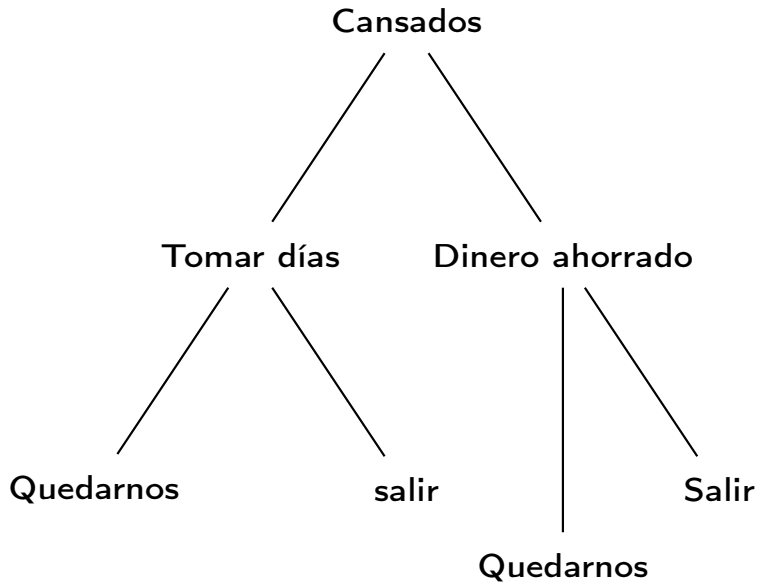
Tomás Fontecilla

9 de septiembre de 2022

Árboles de Decisión - introducción

Los métodos de árboles están basados en las condiciones que requiere la toma de decisión, cortando el espacio paramétrico cada vez que tomamos una decisión y avanzamos a la siguiente etapa.

Así, "Estamos Cansados" divide en dos posibles *outcomes* el espacio, suponiendo que la respuesta sea un "Sí" nos preguntaríamos: "Tenemos dinero ahorrado" lo que nos divide nuevamente en dos y así sucesivamente hasta completar las variables y en cada una tenemos un freno de clasificación final: "Salir de Vacaciones" o "Mejor seguir trabajando".



Árboles de Decisión - CART

Usualmente diríamos que los algoritmos de machine learning están diseñados para tomar decisiones, usualmente haciendo predicciones con el modelo basados en inputs que *nunca se han visto antes* y así, el algoritmo decide una clase a la que asignar. A los modelos de árboles se les llama Classification and Regression Tree, o CART.

intuición - Estructura del árbol

Se usa la base de características para crear continuamente preguntas que se puedan responder con "Sí/No", reduciendo la base de datos hasta aislar los puntos en cada clase.

Árboles de Decisión - de Nodos y otros términos

Como vimos, el modelo sigue una estructura, dividiendo la base de datos cada vez que se responde una de las preguntas.

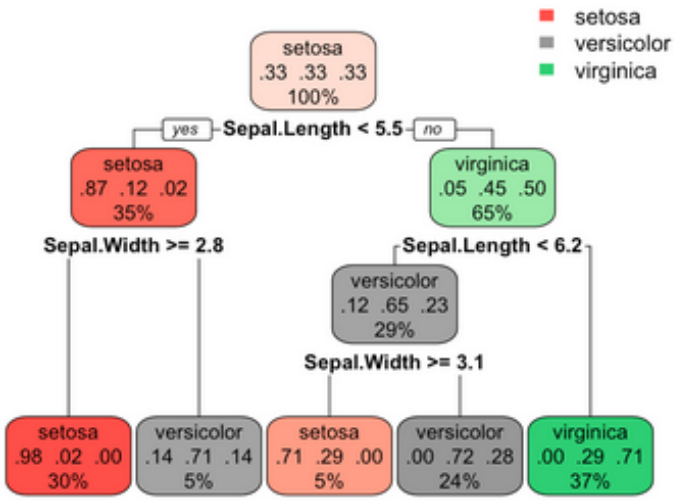
Definiciones

- Se le llama **Nodo** a cada vez que se hace una *pregunta*.
- se llama **nodo raíz** al primer nodo.
- Al cortar un árbol en el proceso, al último nodo se le llama **nodo hoja**.

Al proceso en que se organizan entonces los datos se le conoce como estructura de árbol

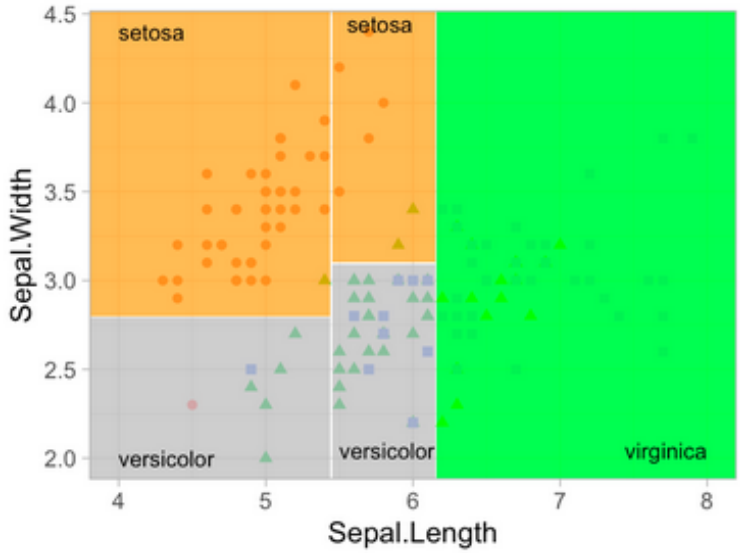
Árboles de Decisión - Estructura

La estructura del árbol entonces es de la siguente forma:



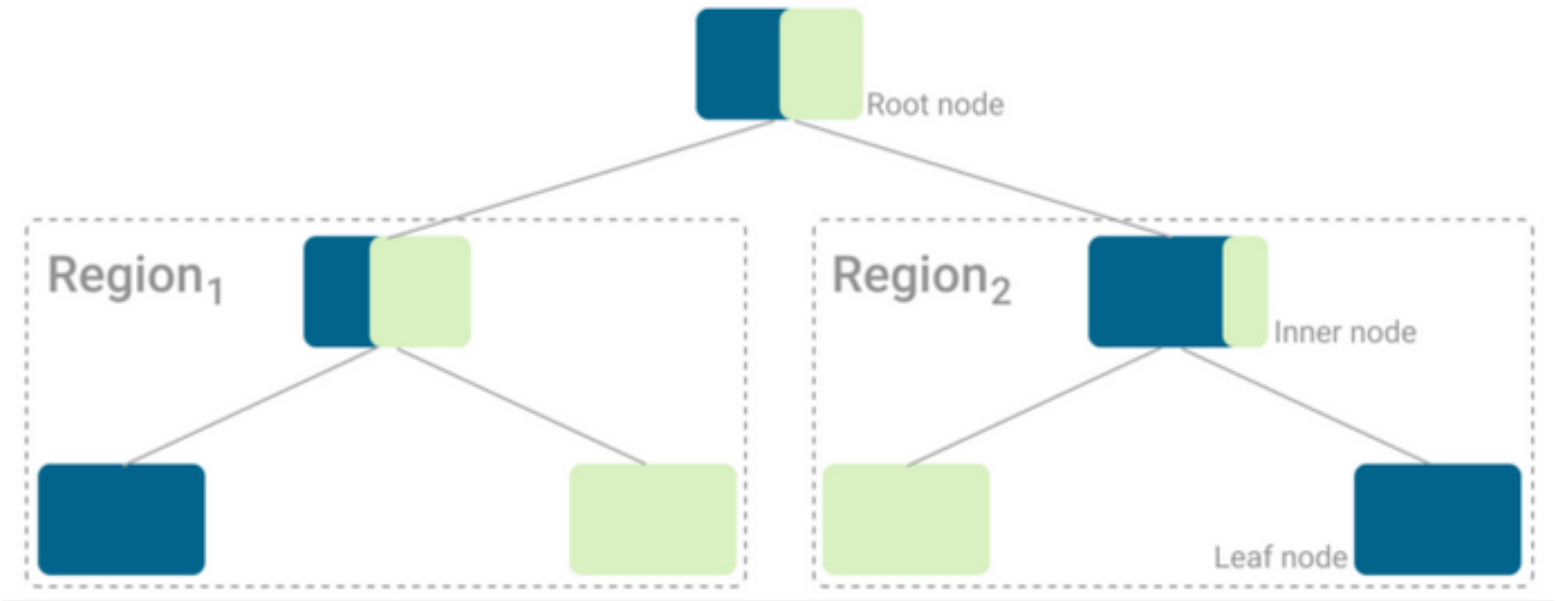
Donde encontramos el valor donde cortamos para decidir (llamado threshold)

Las áreas que cubre la estructura y donde corta, según las variables asociadas al sépalo son las siguientes:



Árboles de Decisión - Intención

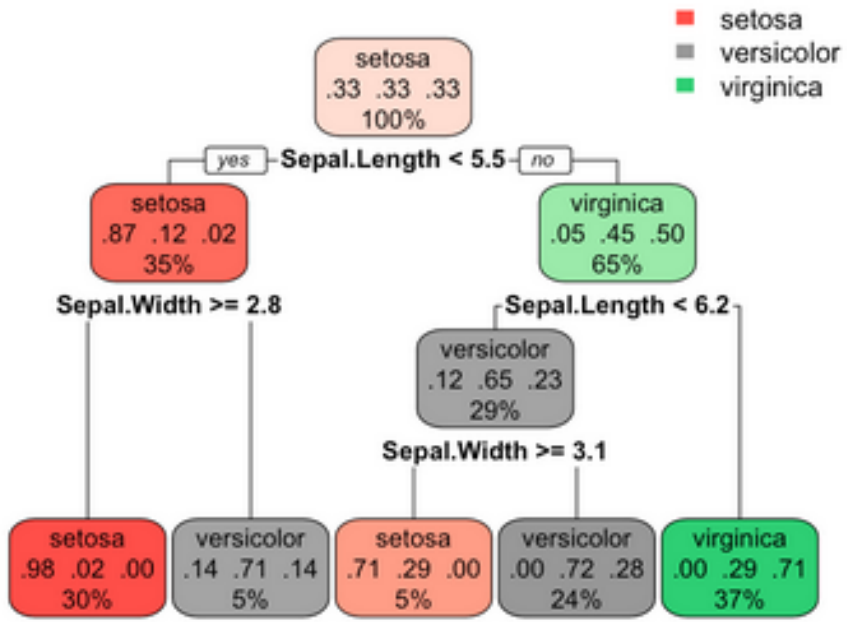
De esta forma, cada rama lleva a una separación de los datos, llamados regiones.



En estas regiones es que el algoritmo está intentando dividir el "dataset" de tal forma que todos los nodos hojas pertenezcan a una única clase.
Éstos son llamados **nodos de hojas puras** o **pure leaf nodes**

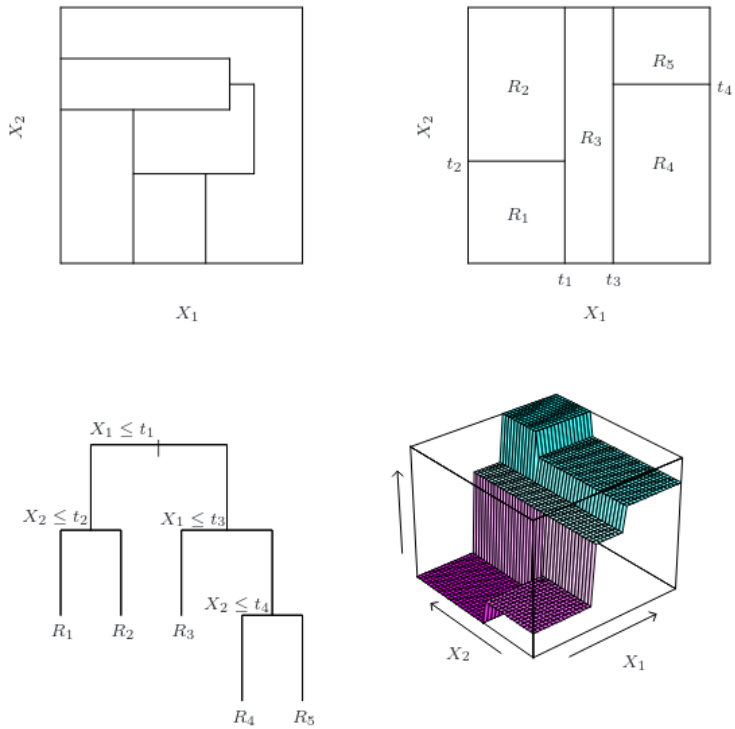
Árboles de Decisión - Clasificación

El problema con el algoritmo es que muchas veces terminamos con **nodos hoja mezclados**, donde no todos los datos tienen la misma clase.
En definitiva el algoritmo puede asignar sólo una clase a un punto en cada nodo hoja.
Por lo tanto, en los nodos hoja mezclados, el algoritmo asigna la **clase más común** a todos los puntos en ese nodo.



Árboles de Decisión - Construcción

Como hemos discutido, los modelos basados en árboles particionan el espacio de características en un conjunto de rectangulos, y luego ajustan un modelo simple (como una constante) en cada uno.
Primero, dividimos el espacio en dos regiones y modelamos la respuesta por el promedio de Y en cada region. Escogemos la variable y *punto de corte* que mejor haga el ajuste.
Luego una o más regiones son divididas nuevamente, se ajusta el modelo, se ve qué variable es la que mejor ajuste y se establece el punto de corte nuevamente, hasta que aplique la regla de parada.



Árboles de Decisión - Modelos de clasificación

Si bien los CART permiten generar regresiones, nos enfocaremos en el modelo de clasificación.
En este caso, siendo la clasificación una clase $p = 1, ..., K$ la forma de corte de las regiones se basa en la proporción de observaciones de la clase k en el nodo m. la podemos escribir de la forma:

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k)$$

Luego, clasificamos las observaciones en el nodo m en la clase $k(m) = argmax_k \hat{p}_{mk}$ o sea, la clase mayoritaria en el nodo m.

Árboles de Decisión - Medidas de impureza

Misclassification error

$$\frac{1}{N_m} \sum_{i \in R_m} I(y_i \neq k(m)) = 1 - \hat{p}_{mk(m)}$$

Índice Gini

$$\sum_{k \neq k'} \hat{p}_{mk} \hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

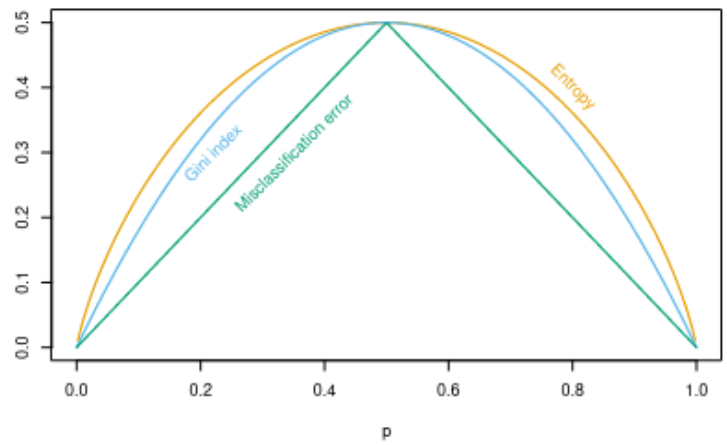
Devianza o entropía cruzada

$$-\sum_{k=1}^{K} \hat{p}_{mk} log\, \hat{p}_{mk}$$

En el caso de dos clases, estas tres medidas serían:

- $1 - max(p, 1 - p)$
- $2p(1 - p)$
- $-plog(p) - (1 - p)log(1 - p)$

El siguiente gráfico muestra estas medidas de impureza como una función de la proporción p en la clase 2. La devianza fue escalada a (0.5,0.5)



Árboles de Decisión - Por qué usarlos

Las principales ventajas:

- Interpretabilidad: Además de ser fácilmente graficables, el pensamiento humano tiende a regirse por la dicotomía de alternativas "SíNo"
- No se requiere preprocesamiento: Mientras varios algoritmos de machine learning requieren que las características sean parecidas y nos obligue a *normalizarlas* antes para obtener mejores resultados, en el caso de árboles de decisión esto no es así.
- Datos robustos: Este algoritmo puede lidiar con varios tipos de datos, sean numéricos o categóricos, sin necesidad de preclasificar las características categóricas.

Las principales desventajas:

- Tienen alta varianza: Dada la naturaleza de crear divisiones y subparticiones de datos, al crear los datos de entrenamiento en 2 tendremos 2 árboles totalmente distintos.
- No son los mejores predictores: Generalmente no tienen el mismo nivel de precisión predictiva que algunos de los otros enfoques de regresión y clasificación.
- Tienden a ser poco robustos: Un pequeño cambio en los datos puede causar un gran cambio en el árbol estimado final. (¡ No confundir con que maneja de forma robusta los datos!)

Random Forest - Motivación

Estás tan cerca del árbol que no ves el bosque

Ya vimos que los árboles de decisión tienen un problema de estabilidad. Esto se debe a que al hacer reiteradas veces el algoritmo, arroja distintos puntos de corte según haga las divisiones.
Según vimos, los árboles tienden a tener alta varianza. Para esto introduciremos dos conceptos que nos permitirán mejorar nuestros análisis:

Bagging

También llamado *boostrap aggregation* es una técnica para reducir la varianza de una función de predicción estimada.

Boosting

Secuencialmente aplica un algoritmo de clasificación débil a versiones modificadas repetidamente de los datos para producir una secuencia de clasificadores débiles. Estas predicciones son combinadas para crear una mayoría de votos ponderado para producir una predicción final.

Adelanto próxima clase: Adaboost

el algoritmo adaboost, uno de los más famosos algoritmos de boosting, es el siguiente:

AdaBoost

1. Inicializa los pesos de las observaciones $\omega_i = \frac{1}{N}, i = 1, ..., N$
2. Para $m = 1$ a M:
   a) Ajusta un clasificador $G_m(x)$ a los datos de entrenamiento usando los pesos ω_i
   b) Calcula
   $$err_m = \frac{\sum_{i=1}^{N} \omega_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} \omega_i}$$
   c) Calcula $\alpha_m = log(\frac{1-err_m}{err_m})$
   d) Fija $\omega_i \leftarrow \omega_i \cdot exp[\alpha_m \cdot I(y_i \neq G_m(x_i))], i = 1, 2, ..., N$
3. Output: $G(x) = sign\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$

Random Forest - Definición

La idea principal entonces es realizar bagging y boosting para generar todo un bosque de árboles de decisión. El algoritmo es como sigue:

Algoritmo de Random Forest

1. Para cada b = 1 a B:
   a) Se obteniene una muestra bootstrap Z^* de tamaño N de la muestra de entrenamiento
   b) Hacer crecer un árbol de bosque aleatorio T_b sobre los datos de bootstrap, recursivamente realizando los siguientes pasos para cada nodo terminal hasta que se alcance el nodo mínimo de tamaño n_{min}:
      i. Seleccionar m variables al azar de las p variables.
      ii. Escoger la mejor variable/Punto de separación entre las m.
      iii. Separar el nodo en dos nodos hijos.
2. se obtiene como *output* el árbol ensamblado $\{T_b\}_1^B$

Random Forest - Predicción

Para hacer una predicción en un nuevo punto x:

Regresión:

$$\hat{f}_{rf}^{B}(x) = \frac{B}{\sum_{b=1}^{B} T_b(x)}$$

Clasificación:

Sea $\hat{C}_b(x)$ la predicción de clase del b-ésimo árbol de random forest. Luego:

$$\hat{C}_{rf}^{B}(x) = \text{ mayoría de voto de } \left\{\hat{C}_b(x)\right\}_1^B$$

Entonces, si bien los árboles de decisión son ideales para hacer *bagging*, nos genera el problema que para mejorar la predicción solo lo podríamos hacer por reducción de varianza (ya que el sesgo se mantiene a lo largo de los árboles), esto es a diferencia de *Boosting* que reduce el sesgo de una forma adaptativa, por lo tanto no identicamente distribuida.

Random Forest - Como lo hace

El promedio de B variables iid con varianza σ^2 tiene varianza $\frac{1}{B}\sigma^2$. si las variables son solo i.d. (no necesariamente independientes), entonces hay que considerar la correlación entre pares ρ y la varianza queda como

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

A medida que B crece, el segundo término desaparece, pero ρ nos limita los beneficios de promediar.
La idea de Random Forest entonces es mejorar la reducción de la varianza de *bagging* al reducir la *correlación* entre árboles, sin incrementar mucho la varianza.

Random Forest

Entonces, al hacer crecer un árbol en el dataset hecho con bootstrap, *Antes de cada división, seleccionamos* $m \leq p$ *de los inputs al azar como candidatos para dividir*

Típicamente, los valores de m son $\sqrt{p}$ o incluso 1.

Después de B árboles creados de esta forma $(\{T(x_i; \Omega_b)\}_1^B)$, el bosque predictor queda:

$$\hat{f}_{rf}^B(x) = \frac{1}{B} \sum_{b=1}^{B} T(x_i; \Omega_b)$$