Programa Técnico Intensivo en Data Science

Decision Trees

Alberto Oteo García
Merck Kgaa
aog1395@gmail.com

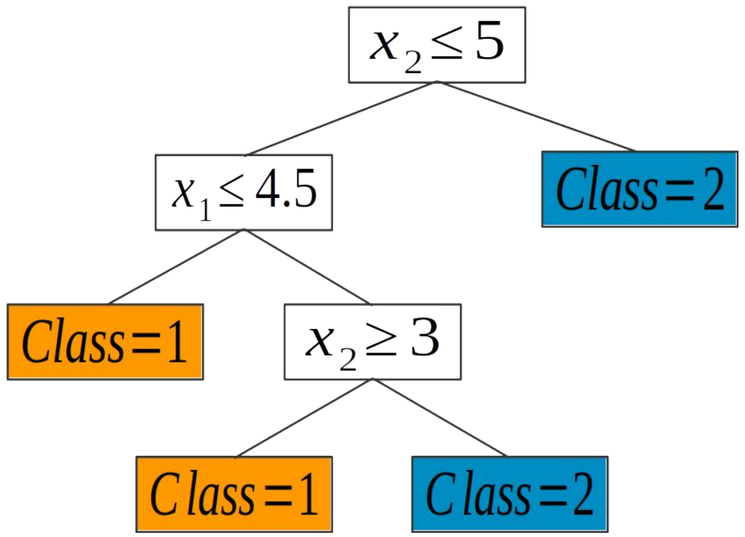
IDAL

Intelligent
Data
Analysis
Laboratory

Datamecum

¿Qué es un árbol de decisión?

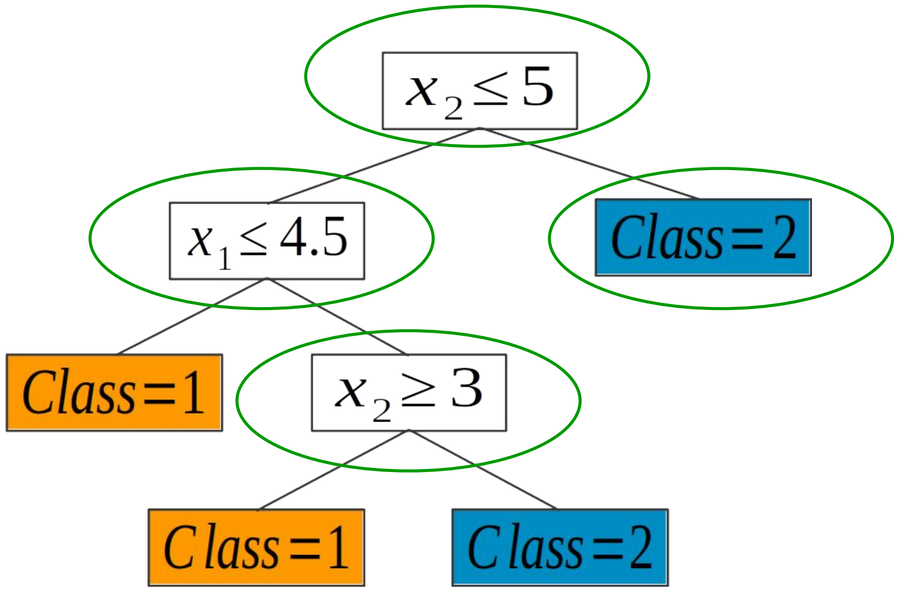
- Un árbol de decisión es una serie de preguntas de si / no sobre los datos input que se introducen en el modelo.



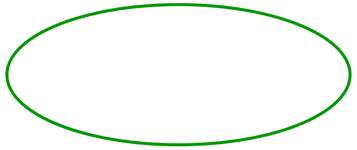
x_1	x_2	y
3.5	2	1
5	2.5	2
1	3	1
2	4	1
4	2	1
6	6	2
2	9	2
4	9	2
5	4	1
3	8	2

¿Qué es un árbol de decisión?

- Un árbol de decisión es una serie de preguntas de si / no sobre los datos input que se introducen en el modelo.



x_1	x_2	y
3.5	2	1
5	2.5	2
1	3	1
2	4	1
4	2	1
6	6	2
2	9	2
4	9	2
5	4	1
3	8	2

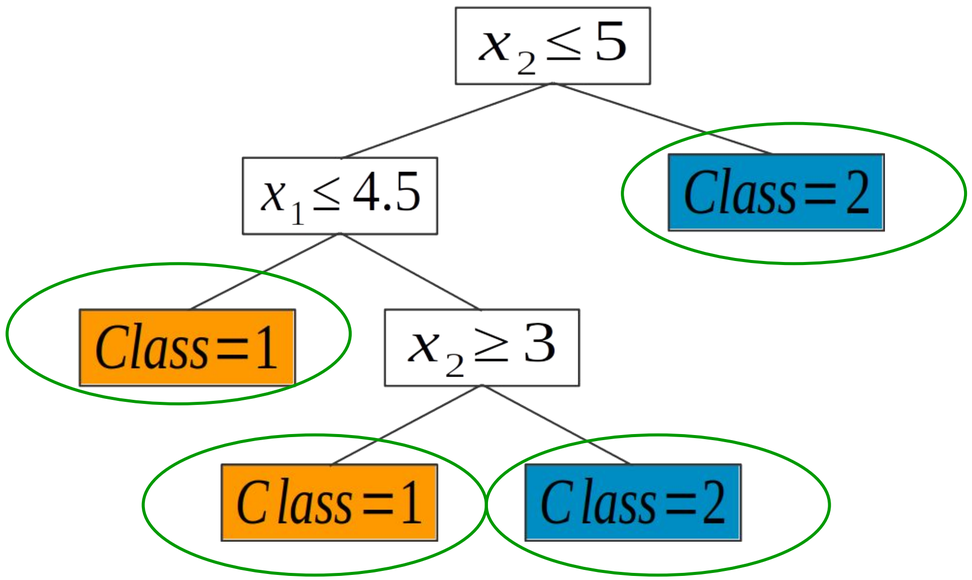


Nodos de partición (split nodes)

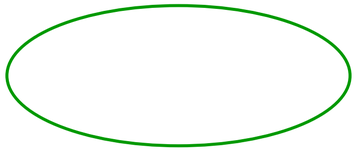
Datamecum

¿Qué es un árbol de decisión?

- Un árbol de decisión es una serie de preguntas de si / no sobre los datos input que se introducen en el modelo.



x_1	x_2	y
3.5	2	1
5	2.5	2
1	3	1
2	4	1
4	2	1
6	6	2
2	9	2
4	9	2
5	4	1
3	8	2

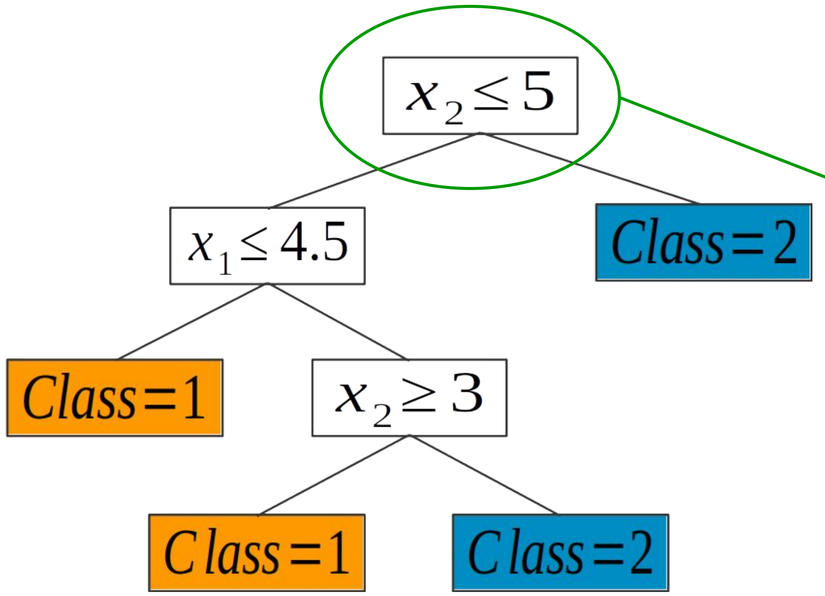


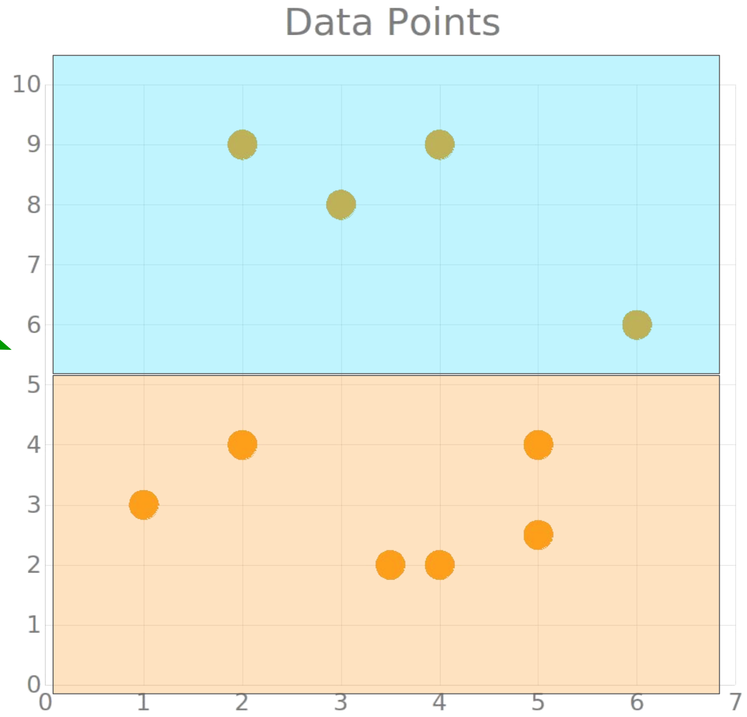
Nodos hoja (leaf nodes)

Datamecum

¿Qué es un árbol de decisión?

- ¿Qué feature (x1 o x2) utilizamos para separar con el objetivo de separar mejor las clases 1 y 2?

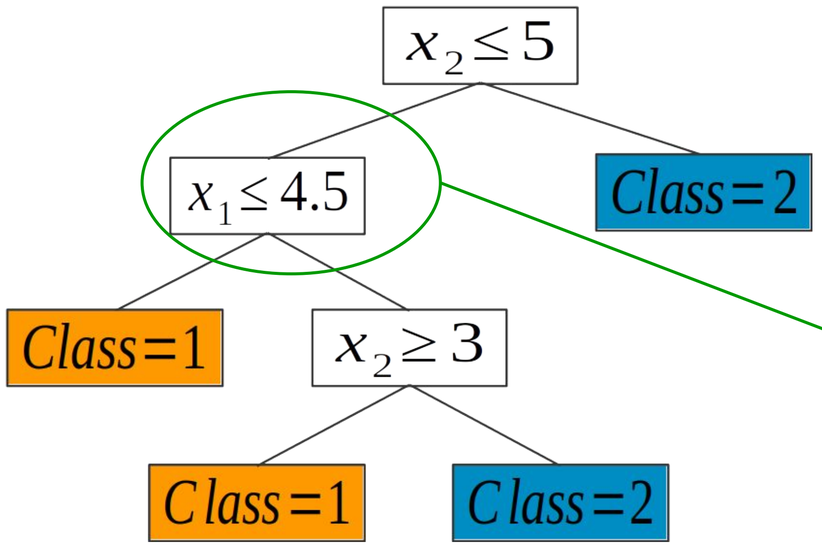


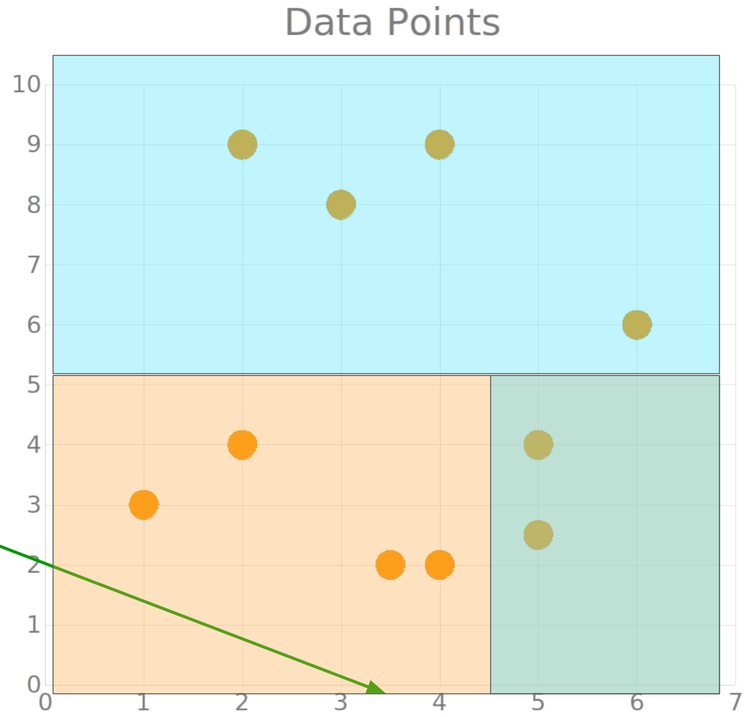


Datamecum

¿Qué es un árbol de decisión?

- ¿Qué feature (x1 o x2) utilizamos para separar con el objetivo de separar mejor las clases 1 y 2?

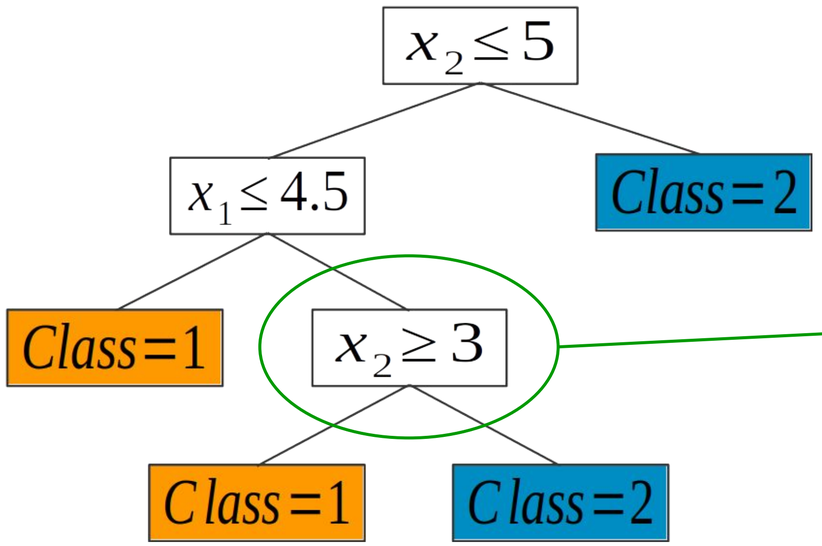


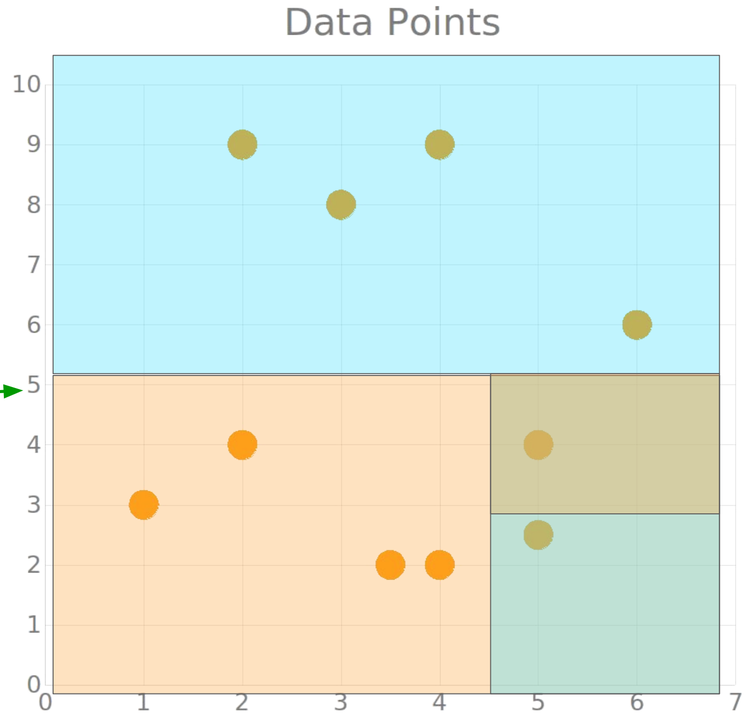


Datamecum

¿Qué es un árbol de decisión?

- ¿Qué feature (x1 o x2) utilizamos para separar con el objetivo de separar mejor las clases 1 y 2?

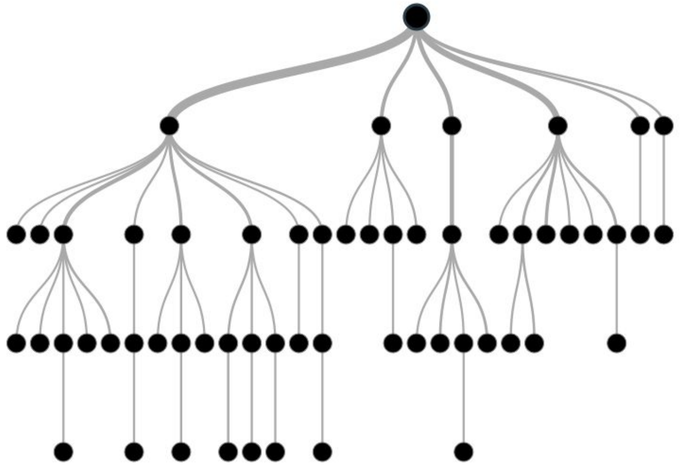




Datamecum

¿Qué es un árbol de decisión?

- El árbol de decisión es un tipo de algoritmo de aprendizaje supervisado
- Tiene una variable objetivo predefinida que se utiliza sobre todo en problemas de clasificación.
- Funciona tanto para variables de entrada como de salida categóricas y continuas.
- En esta técnica, dividimos la población o muestra en dos o más conjuntos homogéneos (o subpoblaciones) basándonos en el divisor/diferenciador más significativo de las variables de entrada.



Ejemplo de árbol de decisión

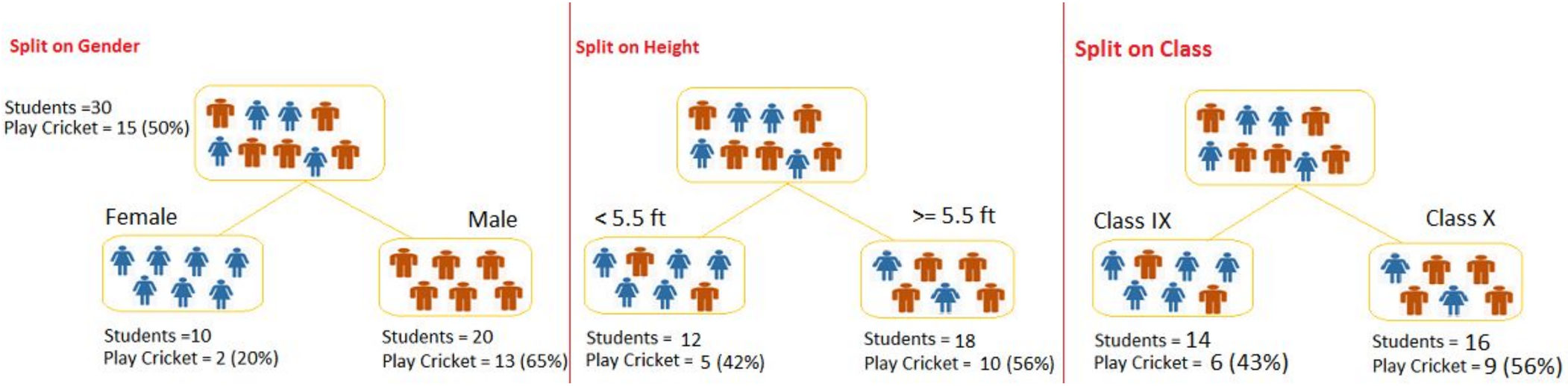
Supongamos que tenemos una muestra de 30 estudiantes con tres variables: sexo (chico/chica), clase (IX/X) y altura (entre 1,70 y 1,80 m).

Género	Clase	Altura
Hombre/Mujer	IX/X	Entre 170-180 cm

15 de estos 30 juegan al críquet en su tiempo libre. En este problema, necesitamos segregar a los estudiantes que juegan al críquet en su tiempo libre basándonos en una variable de entrada altamente significativa entre las tres.

El árbol de decisión, segregará a los estudiantes basándose en todos los valores de las tres variables e identificará la variable que crea los mejores conjuntos homogéneos de estudiantes (que son heterogéneos entre sí).

Ejemplo de árbol de decisión



El árbol de decisión identifica la variable más significativa y su valor que proporciona los mejores conjuntos homogéneos de población. ¿Cómo identifica la variable y la división?

Datamecum

Programa Técnico Intensivo en Data Science

Componentes de los árboles de decisión

Datamecum

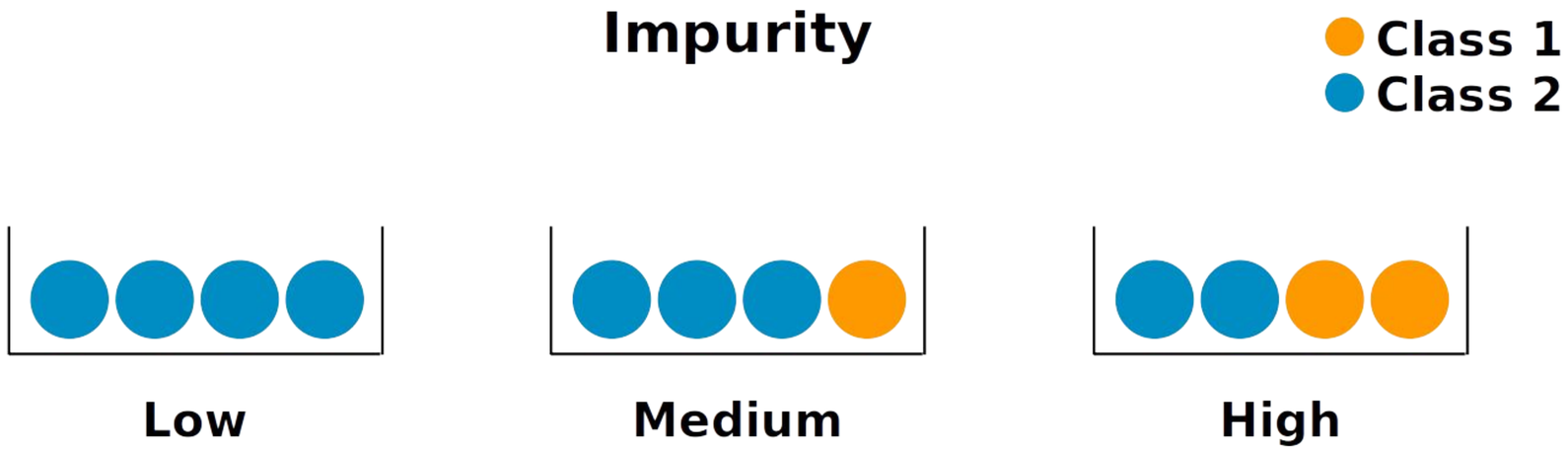
Tipos de árboles de decisión

Árbol de decisión de variable categórica: Árbol de Decisión que tiene una variable objetivo categórica, entonces se llama árbol de decisión de variable categórica. Ejemplo: - En el escenario anterior del problema del estudiante, donde la variable objetivo era "El estudiante jugará al cricket o no", es decir, SÍ o NO.

Árbol de decisión de variable continua: El Árbol de Decisión tiene una variable objetivo continua, entonces se llama Árbol de Decisión de Variable Continua.

Impurity

- La clave de todo el proceso reside en la noción de impureza o imputity.
- Para decidir la partición en cada nodo tenemos que considerar la impureza de sus nodos hijos. Queremos que esta sea lo más baja posible (o más alta posible la pureza).
- Asumamos el siguiente ejemplo a alto nivel.



Impureza en clasificación

- Hay dos opciones comunes:
  - Entropía
  - Gini

- Ambas son muy similares y no tienen significativas diferencias sobre la performance final del modelo.

- Ya que el logaritmo es más complejo a la hora de realizar los cálculos, el índice de Gini es el más recomendado comúnmente.

$$Gini = 1 - \sum_{i=1}^{n} p^2(c_i)$$

$$Entropy = \sum_{i=1}^{n} -p(c_i)log_2(p(c_i))$$

Donde p(ci) es la probabilidad de cada clase en el nodo

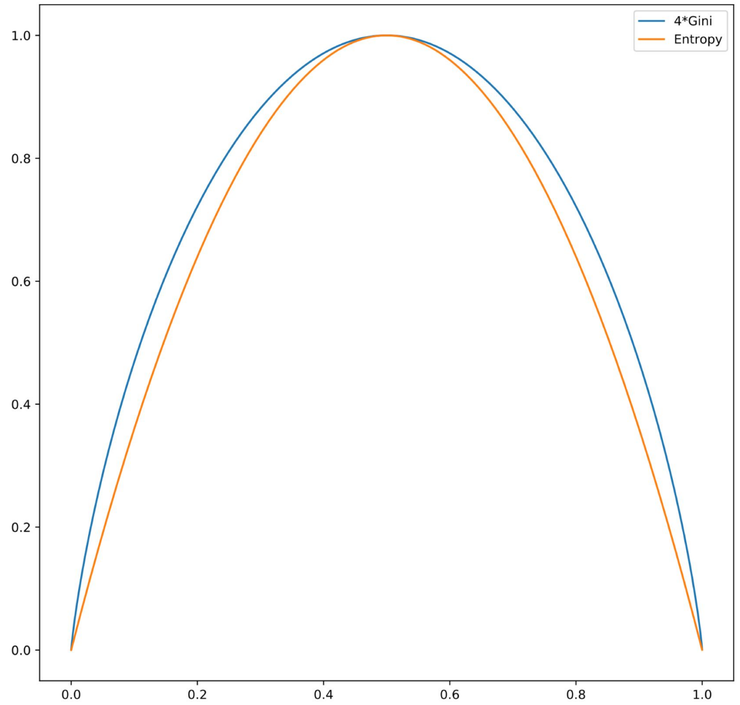
Impureza en clasificación

- Hay dos opciones comunes:
  - Entropía
  - Gini

- Ambas son muy similares y no tienen significativas diferencias sobre la performance final del modelo.

- Ya que el logaritmo es más complejo a la hora de realizar los cálculos, el índice de Gini es el más recomendado comúnmente.

Gini y Entropía para dos clases



$$p_j$$

Impureza en regresión

- Para la regresión, la medida de impureza más simple es la varianza.
- El error absoluto medio también es muy común.
- Al contrario que en modelos basados en el descenso por gradiente, la diferenciabilidad de la función de impureza no es requerida.

Ganancia de información - Information Gain

- Es la diferencia entre la impureza del nodo padre y la suma pesada de las impurezas de los nodos hijos.
- Cuanto más baja la impureza de los nodos hijos, mayor es la ganancia de información.

Utilizando la entropía como medida:

Information Gain = **entropy(**parent**)** – **[average entropy(**children**)]**

candidato	presencia	estudios	experiencia	contratado
1	buena	univ	alta	*si*
2	mala	univ	media	**no**
3	buena	sec	alta	*si*
4	mala	univ	baja	**no**
5	buena	sec	media	*si*
6	buena	univ	media	*si*
7	reg	pri	baja	**no**
8	reg	univ	media	*si*

Nodo padre:

presencia	contratado	no contratado
buena	4	0
mala	0	2
reg	1	1

Entropia: `0.9544340029249649`

```
4/8*Entropia([4/4,0]) + 2/8*Entropia([0,2/2]) + 2/8*Entropia([1/2,1/2]) = 0.25
```

Programa Técnico Intensivo en Data Science

CART

Datamecum

El algoritmo general (CART)

- Asumamos que tenemos una manera de medir el nivel de impureza.
- CART refiere a Classification and Regression Trees. Construye un árbol binario (particiones de sí y no):
    - Tenemos una matriz de datos que refiere a múltiples puntos con múltiples características.
    - Tenemos una etiqueta objetivo que puede ser categórica o numérica.

candidato	presencia	estudios	experiencia	contratado
1	buena	univ	alta	*si*
2	mala	univ	media	**no**
3	buena	sec	alta	*si*
4	mala	univ	baja	**no**
5	buena	sec	media	*si*
6	buena	univ	media	*si*
7	reg	pri	baja	**no**
8	reg	univ	media	*si*

    - Tenemos una medida de impureza que nos dice como de pura es la colección de etiquetas que tenemos.

El algoritmo general (CART)

- **Algoritmo Greedy:** Este espacio de entrada se divide utilizando el método Greedy, que se conoce como binary splitting recursivo. Se trata de un método numérico en el que se alinean todos los valores y se prueban otros puntos de división y se evalúan mediante una función de coste.

- **Criterio de parada:** A medida que avanza por el árbol con los datos de entrenamiento, el método de división binaria recursiva descrito anteriormente debe saber cuándo parar la división. El método de parada más frecuente consiste en utilizar una cantidad mínima de datos de entrenamiento asignados a cada nodo hoja. Si el recuento es inferior al umbral especificado, la división se rechaza y el nodo se considera el último nodo hoja.

- **Poda del árbol:** La complejidad del árbol de decisión se define como el número de divisiones del árbol. Se recomiendan los árboles con menos ramas, ya que son fáciles de comprender y menos propensos a agrupar los datos. El método de poda más rápido y sencillo consiste en examinar cada nodo de hoja del árbol y evaluar el efecto de su eliminación.

- **Preparación de los datos para CART:** El algoritmo CART no requiere una preparación especial de los datos.

El algoritmo general (CART)

Ventajas

- Fácil de entender
- Útil en exploración de datos: identificar la importancia de variables a partir de cientos de variables.
- Menos limpieza de datos: outliers y valores faltantes no influencian el modelo (A un cierto grado)
- El tipo de datos no es una restricción
- Es un método no paramétrico (i.e., no hay suposición acerca del espacio de distribución y la estructura del clasificador)

Desventajas

- Sobre-ajuste
- Pérdida de información al categorizar variables continuas
- Precisión: métodos como SVM y clasificadores tipo ensamblador a menudo tienen tasas de error 30% más bajas que CART (Classification and Regression Trees)
- Inestabilidad: un pequeño cambio en los datos puede modificar ampliamente la estructura del árbol. Por lo tanto, su interpretación no es tan directa como parece.

El algoritmo general (CART)

Limitaciones:

- Overfitting.
- High Variance.
- low bias.
- the tree structure may be unstable.

El algoritmo general (CART)

- Se realiza un bucle sobre cada una de las features y se realiza una partición para cada valor que toma:
  - Se segmenta el dataset en dos partes.
  - Se mantiene un seguimiento sobre la partición que maximiza el descenso medio de impureza (information gain)
  - Recursivamente se pasa sobre los datasets de izquierda y derecha para conseguir nuevos nodos hijos.

- Si algo cumple un criterio de parada ese nodo queda bloqueado y se devuelve el valor más probable asociado con el nodo.
  - Clase mayoritaria o más probable
  - Valor medio de la clase

```
d = 0, endtree = 0
Note(0) = 1, Node(1) = 0, Node(2) = 0
while endtree < 1
    if Node(2^d-1) + Node(2^d) + .... + Node(2^d+1-2) = 2 - 2^d+1
        endtree = 1
    else
        do i = 2^d-1, 2^d, .... , 2^d+1-2
            if Node(i) > -1
                Split tree
            else
                Node(2i+1) = -1
                Node(2i+2) = -1
            end if
        end do
    end if
d = d + 1
end while
```

Programa Técnico Intensivo en Data Science

Regularización en árboles de decisión

Datamecum

¿Qué es la regularización?

- En general la regularización es un método para prevenir el overfitting o ayudar con la optimización.
- En específico, se utilizan términos sobre funciones objetivo para prevenir que estas actúen de maneras “descontroladas”

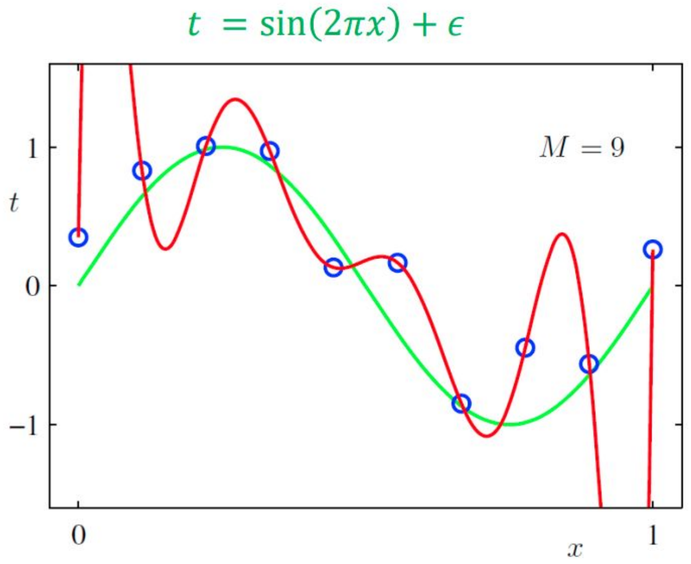


Figure from *Machine Learning and Pattern Recognition*, Bishop

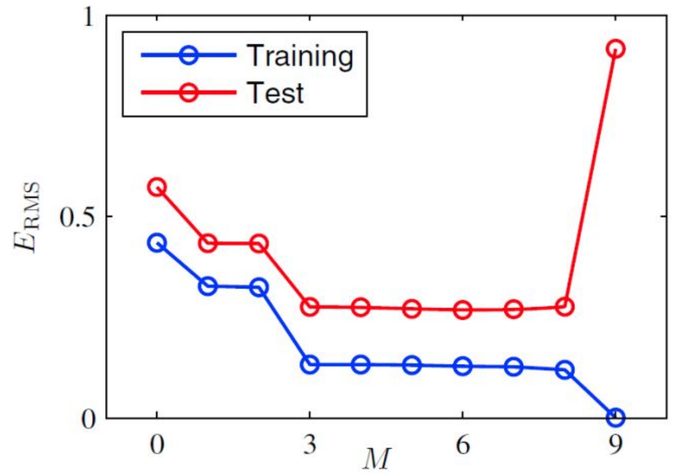


Figure from *Machine Learning and Pattern Recognition*, Bishop

Datamecum

Evitar el overfitting de los árboles

Parada temprana (*early stopping*)

El tamaño final que adquiere un árbol puede controlarse mediante reglas de parada que detengan la división de los nodos dependiendo de si se cumplen o no determinadas condiciones. El nombre de estas condiciones puede variar dependiendo del software o librería empleada, pero suelen estar presentes en todos ellos.

- **Observaciones mínimas para división**: define el número mínimo de observaciones que debe tener un nodo para poder ser dividido. Cuanto mayor el valor, menos flexible es el modelo.
- **Observaciones mínimas de nodo terminal**: define el número mínimo de observaciones que deben tener los nodos terminales. Su efecto es muy similar al de observaciones mínimas para división.
- **Profundidad máxima del árbol**: define la profundidad máxima del árbol, entendiendo por profundidad máxima el número de divisiones de la rama más larga (en sentido descendente) del árbol. Cuanto menor el valor, menos flexible es el modelo.
- **Número máximo de nodos terminales**: define el número máximo de nodos terminales que puede tener el árbol. Una vez alcanzado el límite, se detienen las divisiones. Su efecto es similar al de controlar la profundidad máxima del árbol.
- **Reducción mínima de error**: define la reducción mínima de error que tiene que conseguir una división para que se lleve a cabo.

Todos estos parámetros son lo que se conoce como hiperparámetros porque no se aprenden durante el entrenamiento del modelo. Su valor tiene que ser especificado por el usuario en base a su conocimiento del problema y mediante el uso de estrategias de validación.

Evitar el overfitting de los árboles

Tree pruning

La estrategia de controlar el tamaño del árbol mediante reglas de parada tiene un inconveniente, el árbol crece seleccionando la mejor división en cada momento. Al evaluar las divisiones sin tener en cuenta las que vendrán después, nunca se elige la opción que resulta en el mejor árbol final, a no ser que también sea la que genera en ese momento la mejor división. A este tipo de estrategias se les conoce como *greedy*.

Una alternativa no *greedy* que consigue evitar el *overfitting* consiste en generar árboles grandes, sin condiciones de parada más allá de las necesarias por las limitaciones computacionales, y después podarlos (*pruning*), manteniendo únicamente la estructura robusta que consigue un *test error* bajo.

La selección del *sub-árbol* óptimo puede hacerse mediante *cross-validation*, sin embargo, dado que los árboles crecen lo máximo posible (tienen muchos nodos terminales) no suele ser viable estimar el *test error* de todas las posibles sub-estructuras que se pueden generar. En su lugar, se recurre al *cost complexity pruning* o *weakest link pruning*.

Cost complexity pruning es un método de penalización de tipo *Loss* + *Penalty*, similar al empleado en *ridge regression* o *lasso*. En este caso, se busca el *sub-árbol* T que minimiza la ecuación:

$$\sum_{j=1}^{|T|} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2 + \alpha |T|$$

|T| es el número de nodos terminales del árbol.

Programa Técnico Intensivo en Data Science

Árboles de decisión como parte de algo más

Datamecum

Árboles de decisión y ensembles

- Aunque son excelentes para producir modelos fáciles de entender e implementar, los árboles de decisión también tienden a sobreajustarse a sus datos de entrenamiento, lo que les hace funcionar mal si los datos que se les muestran más tarde no coinciden con los datos con los que se entrenaron.

- En el caso especial de los árboles de regresión, sólo pueden predecir dentro del rango de etiquetas que han visto antes, lo que significa que tienen límites superiores e inferiores explícitos en los números que pueden producir.

- Los métodos ensemble son algoritmos que combinan varios algoritmos en un único modelo predictivo para reducir la varianza, disminuir el sesgo o mejorar las predicciones.

- Los métodos ensemble suelen dividirse en dos categorías:

    - Paralelos: Un método ensemble en el que los modelos que componen los bloques de construcción de los métodos más grandes se generan independientemente unos de otros (es decir, se pueden entrenar/generar como problemas trivialmente paralelos aplicados al conjunto de datos).

    - Secuencial: Un conjunto de métodos en el que los aprendices se generan en un orden secuencial y dependen unos de otros (es decir, sólo pueden entrenarse de uno en uno, ya que el siguiente modelo necesitará información del entrenamiento anterior).

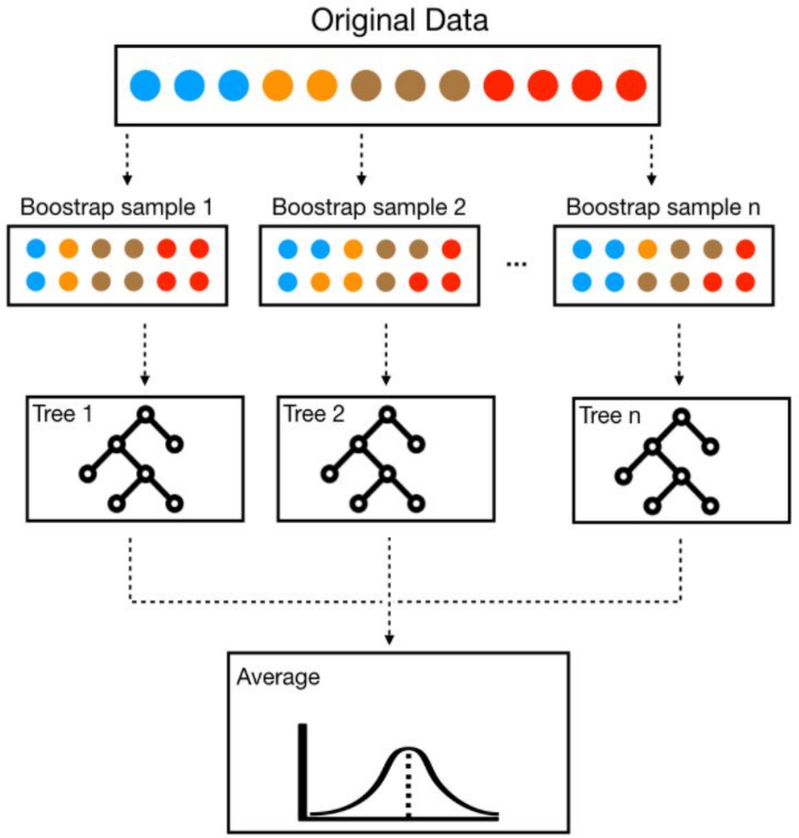
- El algoritmo random forest se basa en un método de ensamblaje paralelo llamado "bagging" para generar sus clasificadores débiles.

Bagging : Bootstrap Aggregating

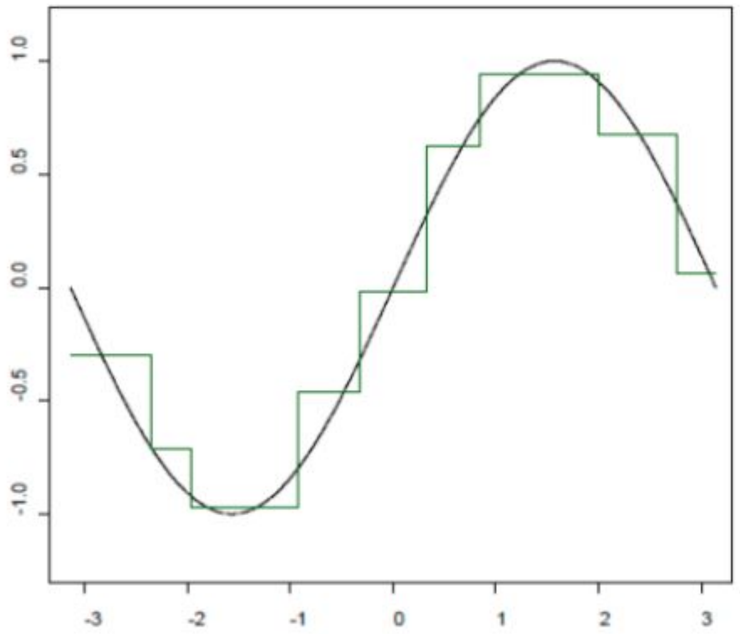
1. Muestreo de registros con sustitución (también conocido como "bootstrap" de los datos de entrenamiento) El muestreo es el proceso de selección de un subconjunto de elementos de una amplia colección de elementos. Bootstrap = Muestreo con sustitución. Significa que un punto de datos de una muestra extraída puede reaparecer también en futuras muestras extraídas.

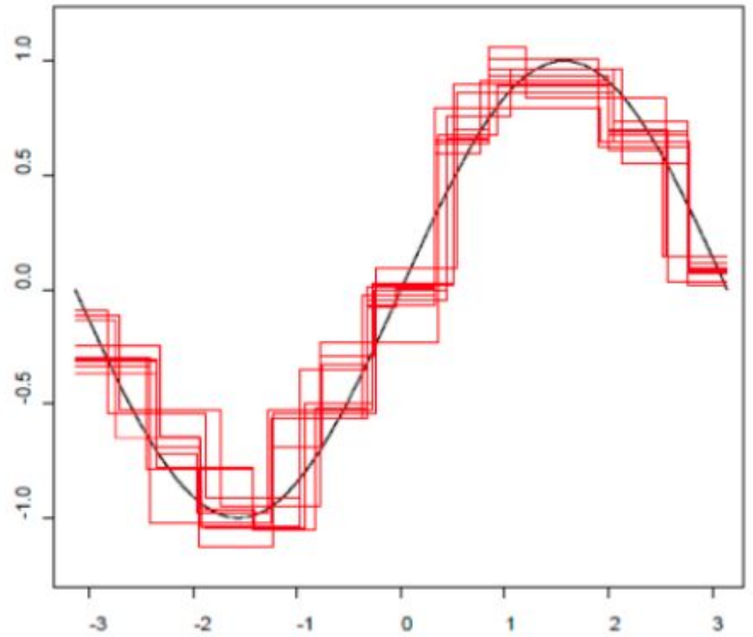
2. Ajustar un árbol de decision a cada conjunto de datos remuestreados.

3. Average predictions



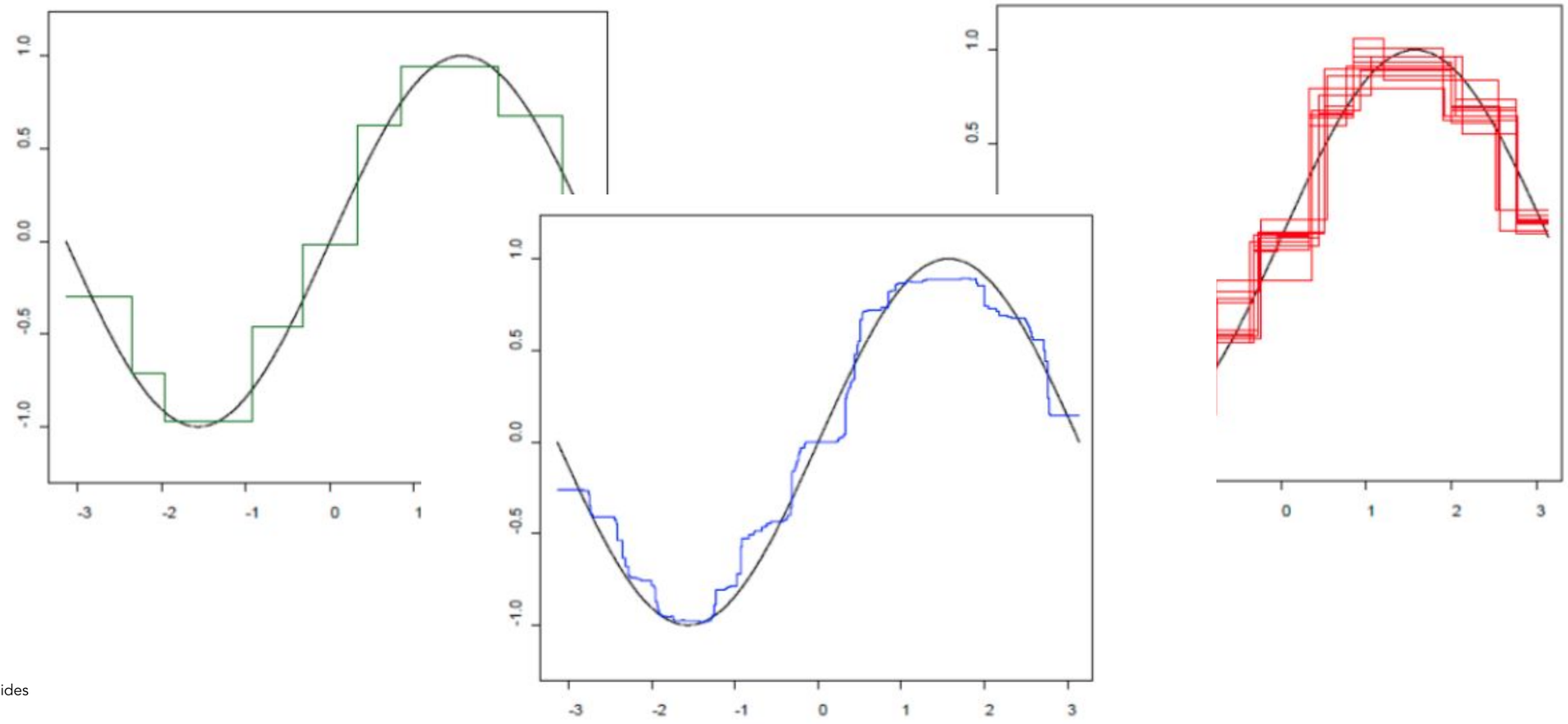
Regresión árboles de decisión





images: Adele Cutler's slides

Regresión Media de 10 árboles



images: Adele Cutler's slides

Datamecum

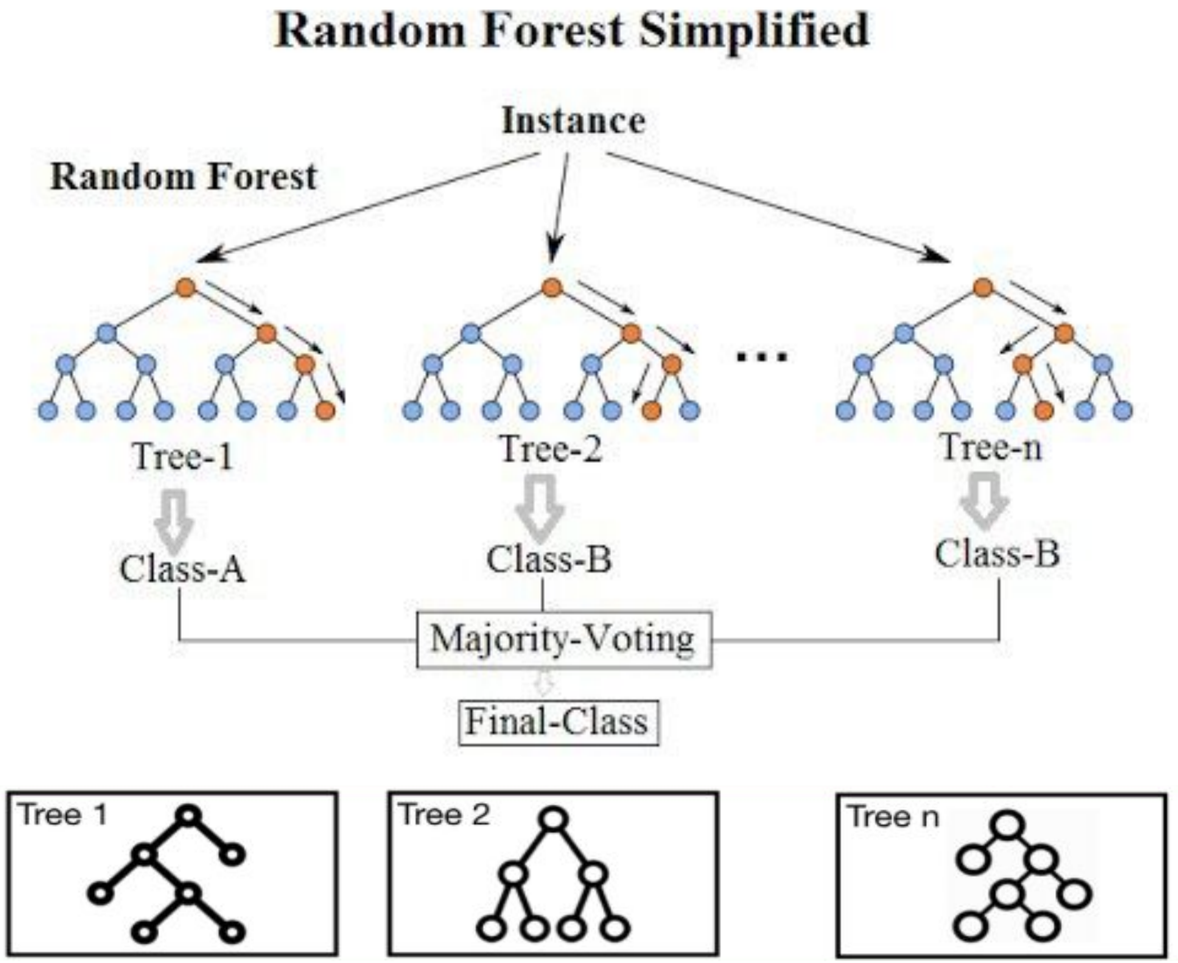
Random Forest

- El random forest se identifica como una colección de árboles de decisión. Cada árbol estima una clasificación, que se denomina "voto". Lo ideal es tener en cuenta cada voto de cada árbol y elegir la clasificación más votada (Votación por mayoría).

- Los Random Forest siguen el mismo proceso de bagging que los árboles de decisión pero cada vez que se va a realizar una división, la búsqueda de la variable de división se limita a un subconjunto aleatorio de m de los p atributos (variables o características) aka Split-Attribute Randomization:

  - árboles de clasificación: m = √p
  - árboles de regresión: m = p/3
  - m se denomina comúnmente mtry

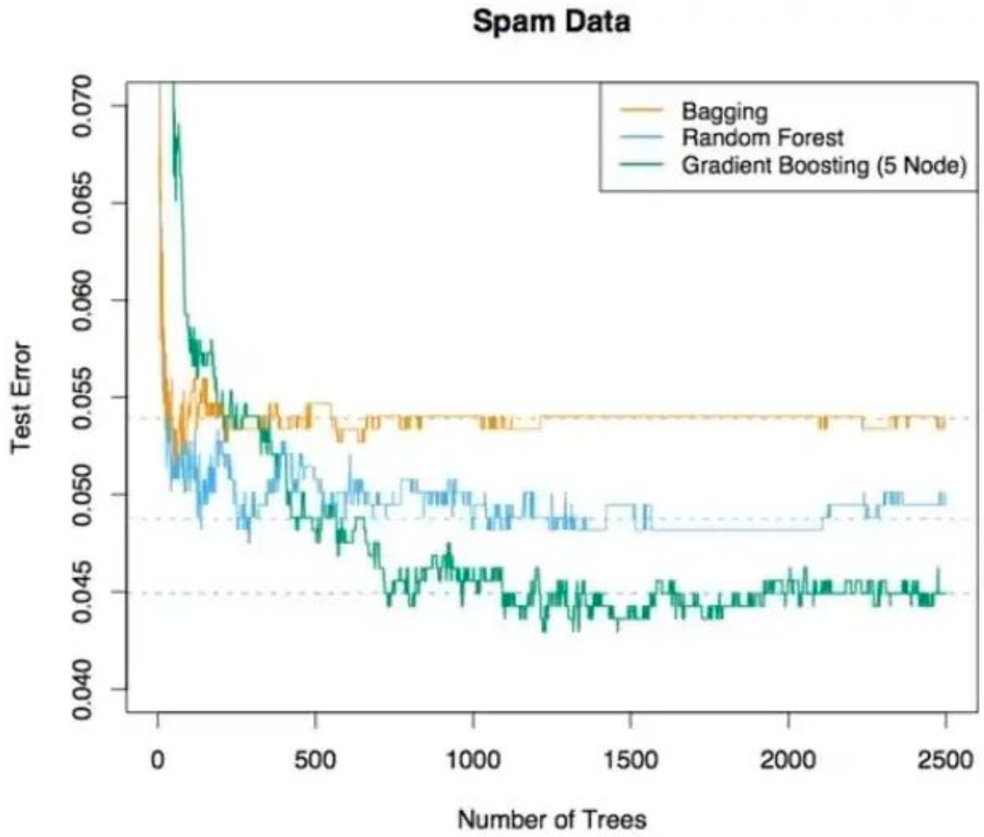
- Random Forests produce many unique trees.



images: Adele Cutler's slides

Random Forest Vs Bagging

- El bagging introduce aleatoriedad en las filas de los datos
- El random forest introduce aleatoriedad en las filas y colur de los datos.
- Combinados, proporcionan un conjunto de árboles más diverso que casi siempre reduce nuestro error de predicci



Source: Professor Trevor Hastie and Professor Robert Tibshirani, Stanford University

Ventajas y desventajas

Ventajas:

- Performance competitiva
- Muy poco ajuste requerido
- Menor tendencia al sobre-ajuste que otros métodos
- Robusto a outliers
- Puede lidiar con valores faltantes
- Provee de un método intrínseco para la selección de características
- Mínimos requisitos de preprocesado

Desventajas:

- Aunque tiene un buen nivel de precisión, habitualmente no puede competir con otros algoritmos de boosting.
- Puede volverse lento en grandes datasets
- Pierde interpretabilidad respecto a los modelos más sencillos.