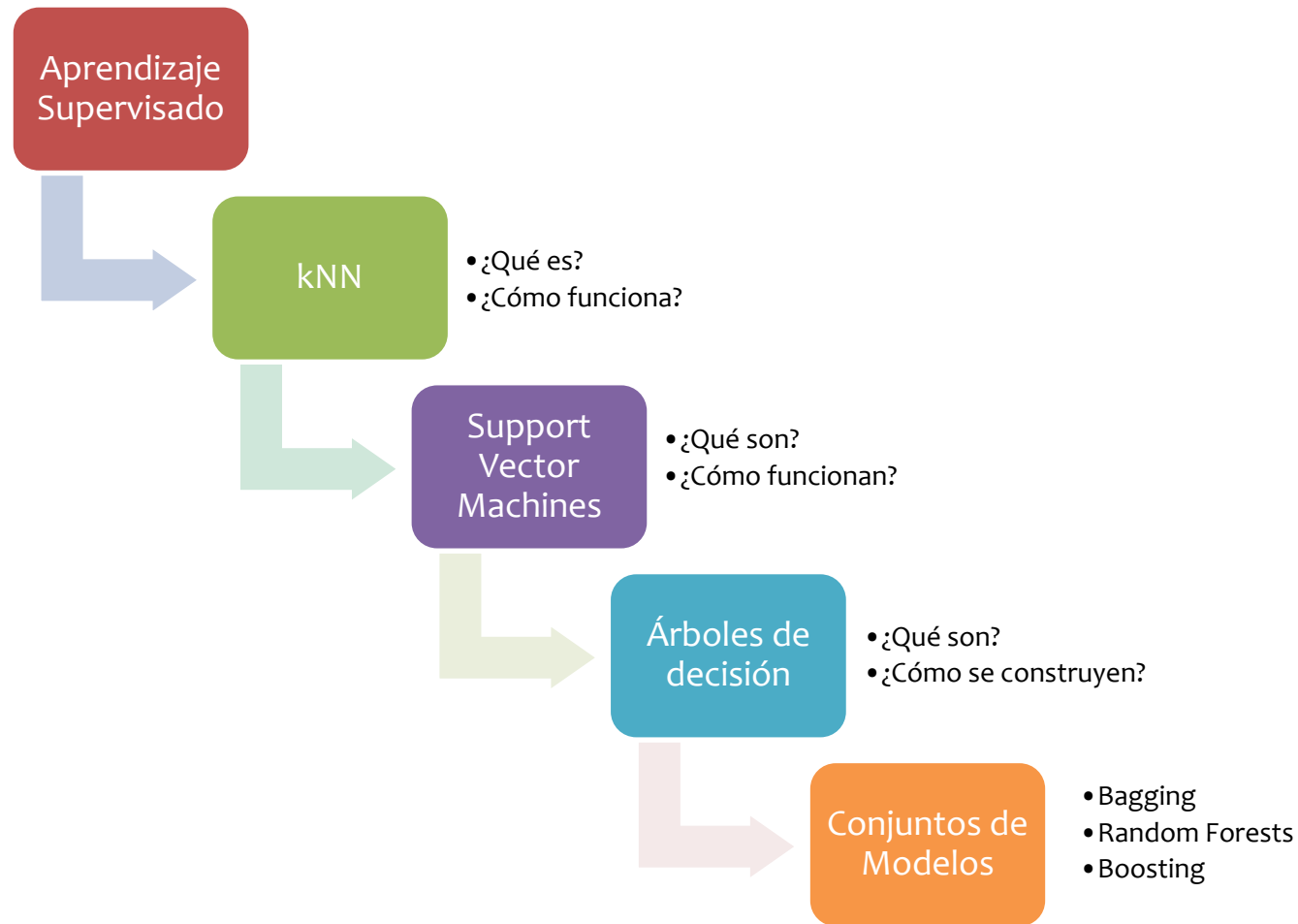


07MBID: Machine Learning VC2: Aprendizaje Supervisado



APRENDIZAJE SUPERVISADO

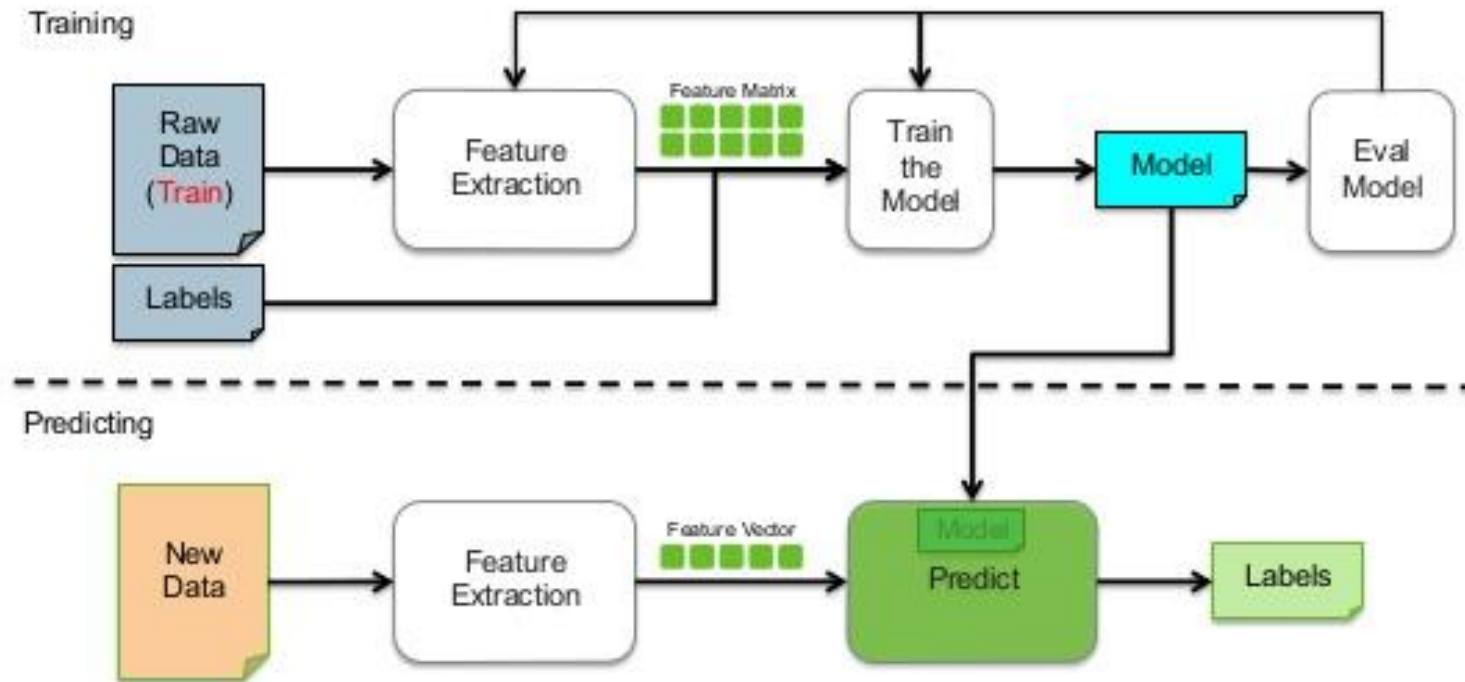
viu | **Universidad**
Internacional
de Valencia



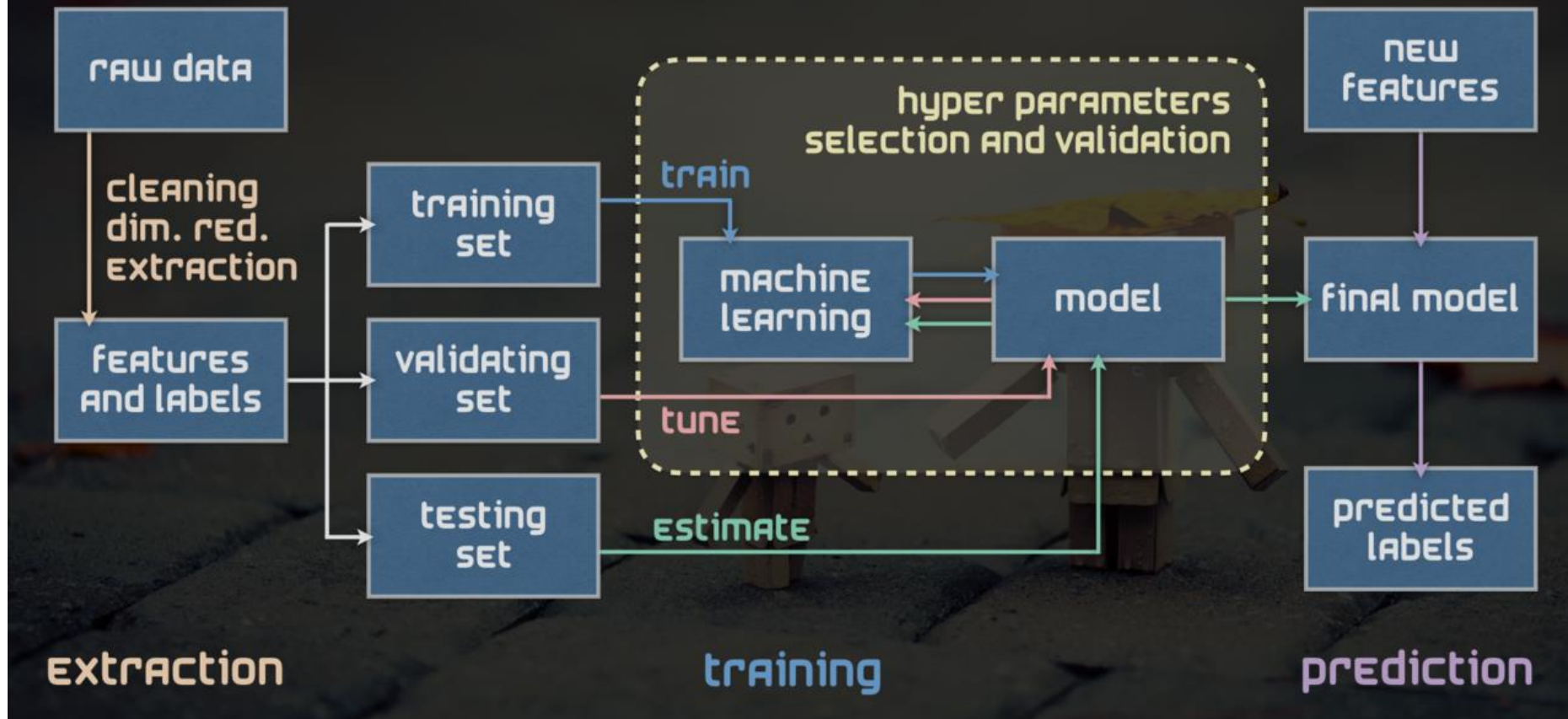
viu | **Universidad**
Internacional
de Valencia

- El principal objetivo del aprendizaje supervisado es inferir un modelo de predicción a partir de un conjunto de datos de entrenamiento que contiene **ejemplos previamente etiquetados**, es decir, instancias sobre las que conocemos la respuesta del problema a resolver.
- De esta manera se puede construir un **modelo** que sirva para hacer **predicciones** a partir de **datos aún no existentes** y/o futuros.
- El aprendizaje supervisado también recibe el nombre de **aproximación de funciones** ya que ante unos datos en los que conocemos tanto la entrada como la salida del proceso, se inferirá la función que dé lugar a la segunda a partir de los primeros.
- Se distinguen dos tipos de tareas: **clasificación y regresión**
- **Necesitaremos conjuntos de datos de entrenamiento y de test**

Supervised Learning Workflow



supervised learning workflow



- La **clasificación** es aquella subcategoría del aprendizaje supervisado en la que el objetivo es predecir **la categoría** o categorías a las que pertenecen nuevas instancias basándonos en los ejemplos de entrenamiento. En este caso la etiqueta que acompaña a cada ejemplo, y que es necesario predecir, es **discreta**, es decir, valores finitos sin necesidad de tener una relación de orden entre ellos, por ejemplo: spam/no spam, bueno/regular/malo, rojo/amarillo, verde; etc.
- El **análisis de regresión** es la parte del aprendizaje supervisado que se ocupa de la predicción de valores numéricos **continuos**, por ejemplo, cuando a partir de variables como la superficie de una casa, y el número de habitaciones es posible predecir su precio. En este caso, el objetivo del algoritmo es inferir las relaciones entre las variables, que son previamente conocidas y que permiten ofrecer una predicción sobre la salida requerida.

- **Validación:** Porcentaje de elementos clasificados correctamente, o también, se puede calcular el cociente de error o de malas clasificaciones. Las herramientas más utilizadas son la matriz de confusión y la curva ROC

	Clasificado Verdadero	Clasificado Falso
Es Verdadero	verdaderos positivos	falsos negativos
Es Falso	falsos positivos	verdaderos negativos

sensitivity

$$= \frac{\text{truepositive}}{\text{totalpositive}}$$

specificity

$$= \frac{\text{truenegative}}{\text{totalnegative}}$$

precision

$$= \frac{\text{verdadero positivo}}{\text{verdadero positivo} + \text{falso positivo}}$$

AUC=1

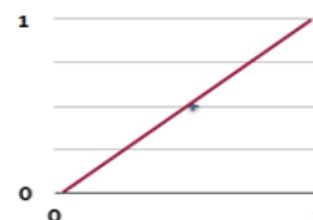
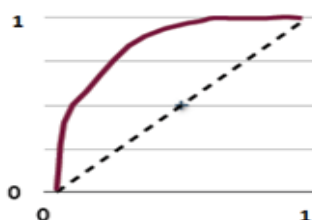
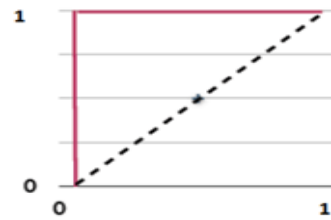
AUC=0,8

AUC=0,5

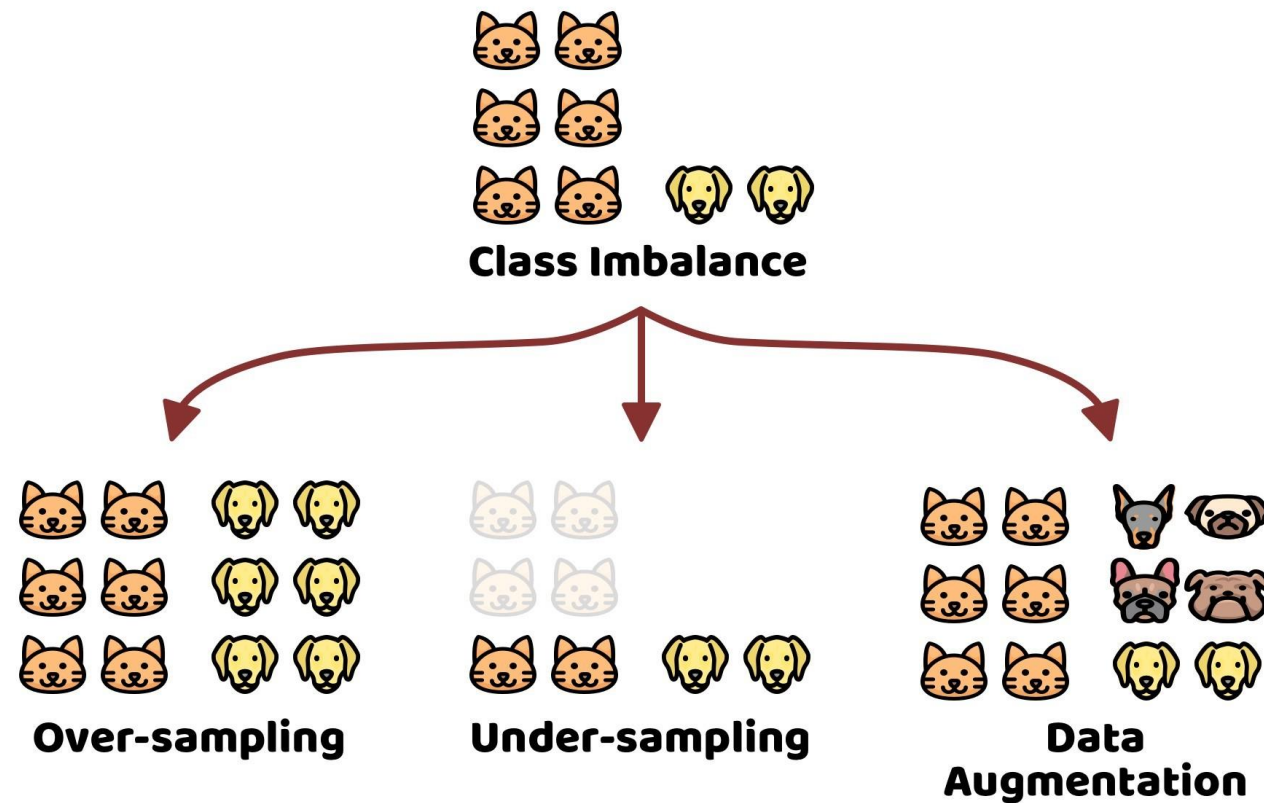
+ valor diagnóstico perfecto

+ valor diagnóstico

+ sin valor diagnóstico



- Si los posibles valores de las variables de salida no están representados de forma uniforme en el conjunto de aprendizaje, los modelos podrían no generalizar bien.
- En ese caso, se debería realizar previamente un “balanceo de clases”.

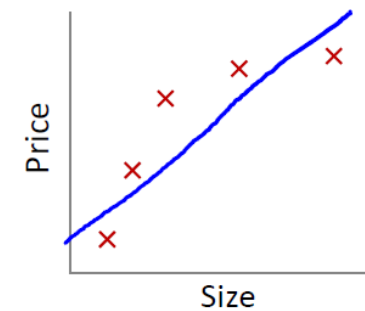


- **Métricas de error:** Para validar problemas de regresión se utilizan métricas de error en los problemas de regresión se utilizan **métricas de error:** MAE, MSE, RMSE, etc.

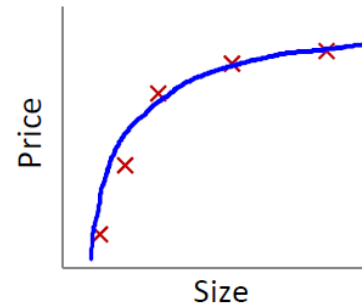
Available metrics

Metric	Definition
Mean Squared Error (MSE)	$MSE = \frac{\sum_{i=0}^{N-1} (y_i - \hat{y}_i)^2}{N}$
Root Mean Squared Error (RMSE)	$RMSE = \sqrt{\frac{\sum_{i=0}^{N-1} (y_i - \hat{y}_i)^2}{N}}$
Mean Absolutue Error (MAE)	$MAE = \sum_{i=0}^{N-1} y_i - \hat{y}_i $
Coefficient of Determination (R^2)	$R^2 = 1 - \frac{MSE}{\text{VAR}(\mathbf{y}) \cdot (N-1)} = 1 - \frac{\sum_{i=0}^{N-1} (y_i - \hat{y}_i)^2}{\sum_{i=0}^{N-1} (y_i - \bar{y})^2}$
Explained Variance	$1 - \frac{\text{VAR}(\mathbf{y} - \hat{\mathbf{y}})}{\text{VAR}(\mathbf{y})}$

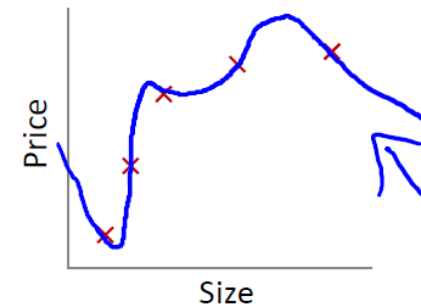
- Cuando se evalúa la calidad de un modelo o un ajuste, es importante medir el error tanto el conjunto de entrenamiento como en la predicción. La utilización exclusiva del error del conjunto de entrenamiento puede conducir a resultados engañosos: **sobreaprendizaje u overfitting**



$\rightarrow \theta_0 + \theta_1 x$
"Underfit" "High bias"

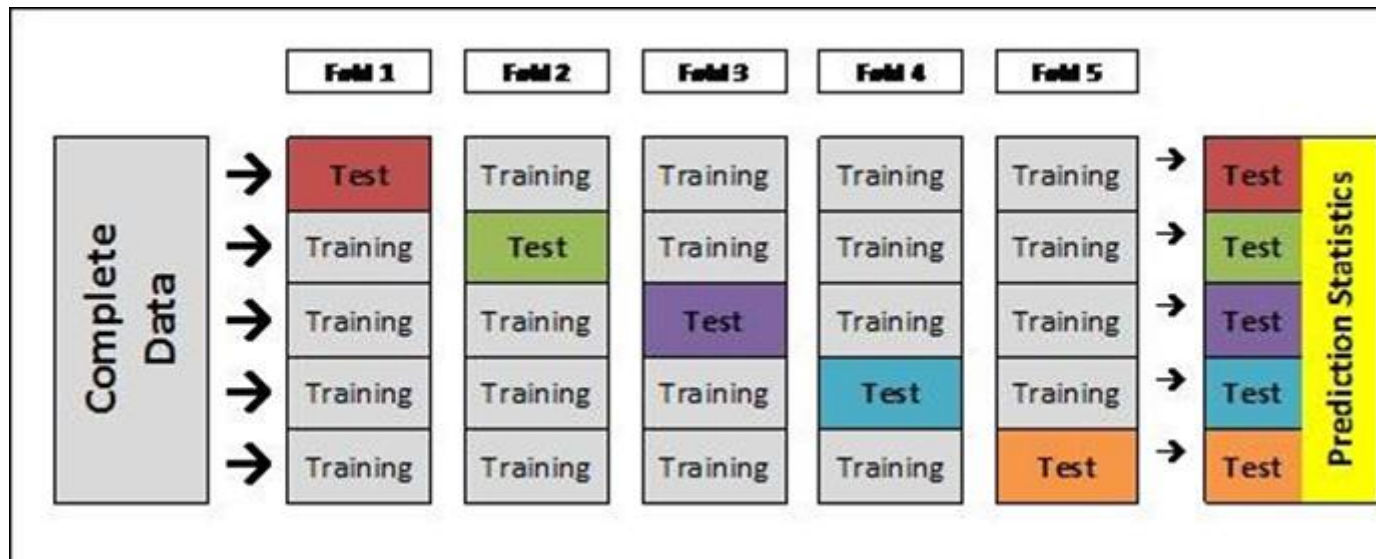


$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2$
"Just right"



$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$
"Overfit" "High variance"

- Para resolver esta cuestión se utiliza la validación cruzada (**Cross-validation**). En este caso, se divide el conjunto de datos originales, apartando pequeños subgrupo de los mismos seleccionados aleatoriamente. Una vez realizado el ajuste, se utilizan los parámetros obtenidos para hallar el valor estimado de la salida de estos datos.

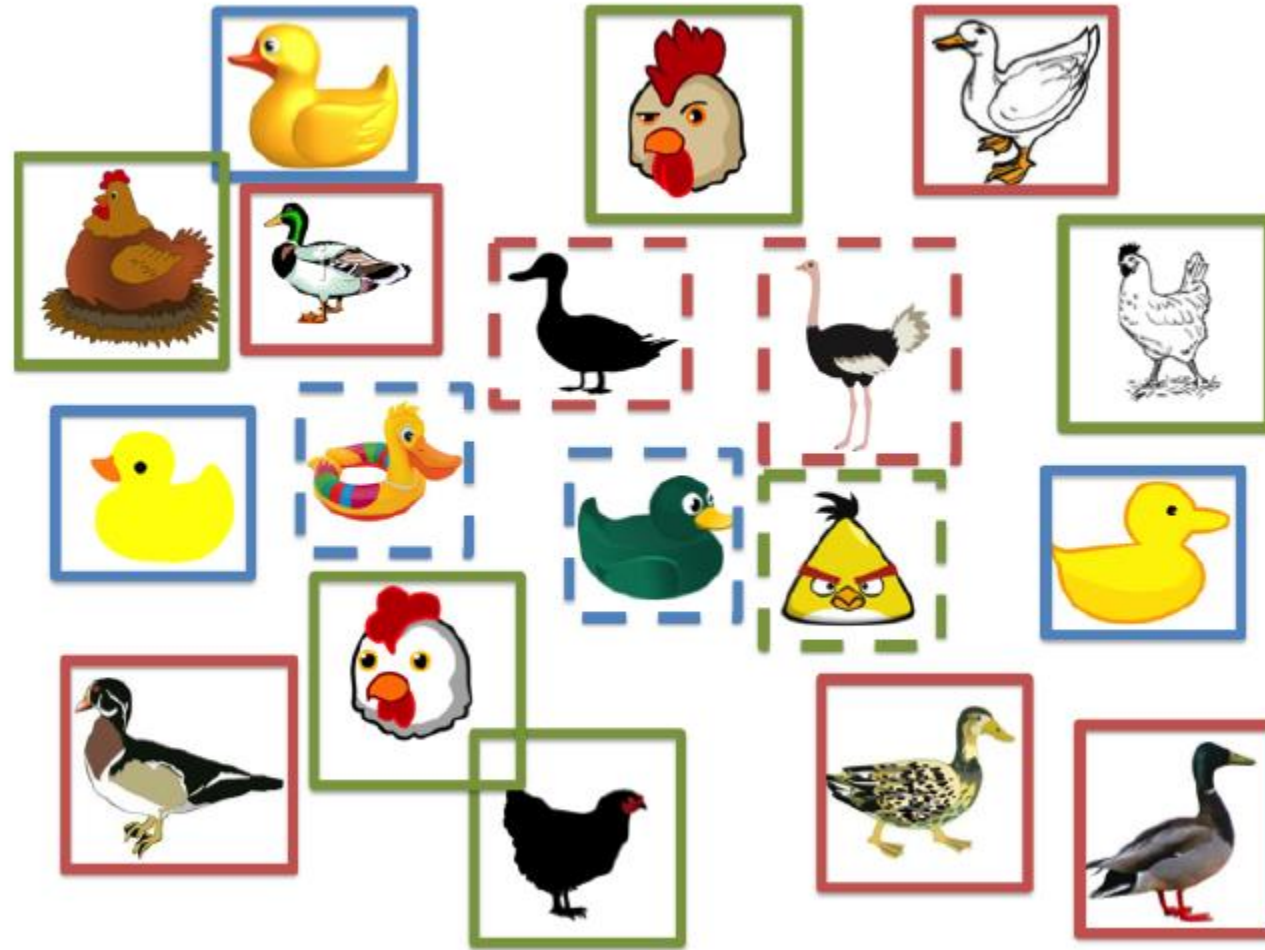


kNN

viu | **Universidad**
Internacional
de Valencia

viu | **Universidad**
Internacional
de Valencia

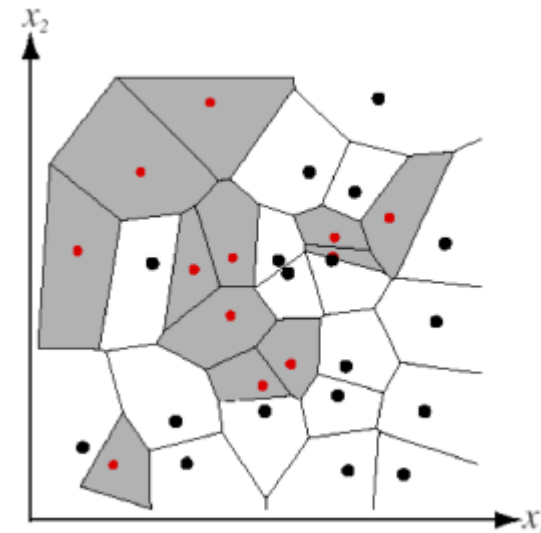
Los algoritmos de los vecinos más cercanos están basados en el aprendizaje por analogía



- Se encuadran en el paradigma perezoso de aprendizaje:
 - **Perezoso:**
 - El trabajo se retrasa todo lo posible,
 - no se construye ningún modelo, el modelo es la propia BD o conjunto de entrenamiento.
 - Más que de entrenamiento hablamos de indexación
 - **Se trabaja cuando llega un nuevo caso a clasificar:**
 - Se buscan los casos más parecidos y la clasificación se construye en función de la clase a la que dichos casos pertenecen.
- **Los algoritmos más conocidos están basados en la regla del vecino más próximo (NN o 1-NN)**

- La técnica más popular del Instance Based Learning es la Regla del vecino más próximo o **Nearest-Neighbour** (NN):
- Si tenemos m instancias de ejemplos en nuestra base de datos, entonces para clasificar un nuevo ejemplo, el procedimiento será el siguiente utilizando una función de distancia d :

1. $c_{min} = clase(e_1)$
2. $d_{min} = d(e_1, e')$
3. para $i=2$ hasta m hacer
 $d = d(e_i, e')$
 si $(d < d_{min})$
 entonces
 $c_{min} = clase(e_i), d_{min} = d$
4. Devolver c_{min} como clasificación de e'




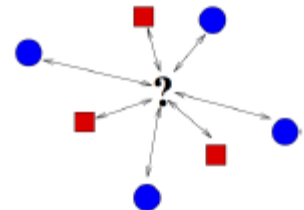
- En el caso de variables nominales se usa la distancia de Hamming:
- Las variables numéricas suelen ser normalizadas al intervalo $[0,1]$
- En este caso algunas de las funciones de distancias más usadas son las de la familia de distancias de Minkovsky (Manhattan, Euclídea, etc.)

$$d_e(e_1, e_2) = \sqrt{\sum_i (e_1^i - e_2^i)^2 + \sum_j d_h(e_1^j, e_2^j)}$$

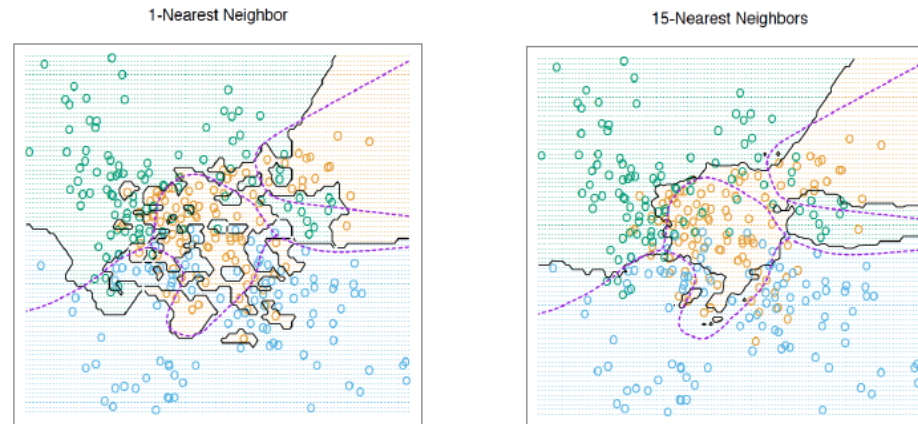
numéricas nominales

- Tratamiento de valores desconocido: la distancia asociada a la j-ésima componente es la máxima: 1.

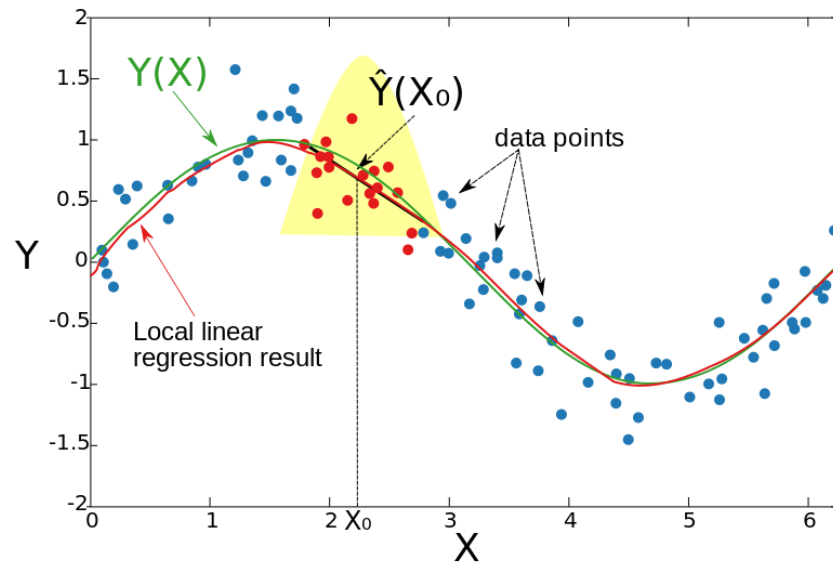
- La extensión de la regla del vecino más próximo, es considerar los k vecinos más próximos: **kNN** (NN se conoce como 1NN)
- Dado el ejemplo a clasificar, la idea es:
 - Seleccionar los **k ejemplos más próximos** al ejemplo
 - Devolver la **clase** que **más se repite** en el conjunto
- Por ejemplo, si $k = 7$ el siguiente caso (?) sería clasificado como 
- Podría pensarse en tratar de forma diferente a los k vecinos, p.e., dependiendo de la distancia al objeto a clasificar, así tendríamos:
- Clasificación: voto por la **mayoría**.
- Clasificación: voto **ponderado** por la (inversa de la) distancia



- Es **robusto** al ruido cuando se usan valores de k moderados ($k > 1$)
- Es bastante **eficaz**, puesto que se pueden usar varias funciones lineales locales para aproximar la función objetivo.
- **Problemas:**
 - Es **ineficiente** en **memoria** ya que hay que guardar toda la BD.
 - Su **complejidad temporal** (para evaluar un ejemplo) es $O(d(n)N)$ siendo $O(d(n))$ la complejidad de la distancia usada sobre vectores de longitud n .



- **kNN** es una técnica **versátil**, puede aplicarse tanto a **clasificación** como a problemas de **regresión**.
- Dados los **k** vecinos usamos la media para predecir.
- Se puede **ponderar** por la distancia, ponderar los atributos, etc.
- El **valor** seleccionado para **k** juega un papel **fundamental**:

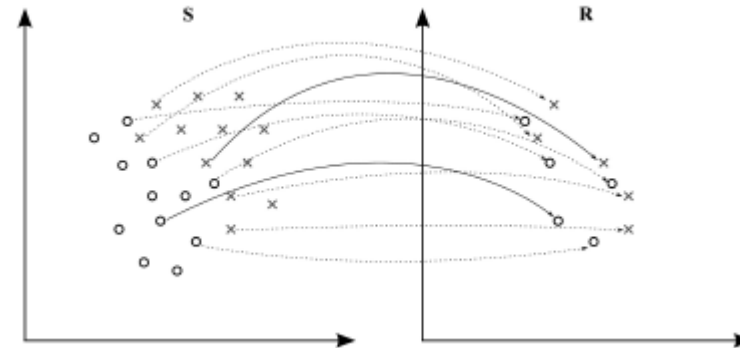


- Hay distintas formas de **mejorar** la eficiencia/eficacia realizando cierto **aprendizaje previo** a la fase de clasificación.
- Una crítica a la regla NN es que **todas las variables se consideran con igual importancia**.
 - **Solución:** pesar los atributos, asociando un peso w_i al atributo i -ésimo que pondere su importancia dentro del contexto:
 - **Pero:** hay que aprender/calcular/optimizar los pesos.

- **Ajuste del valor de k**
 - Si **k es pequeño** habrá un problema de **sensibilidad** al ruido.
 - Si **k es demasiado grande**, las **fronteras** de decisión serán **dispersas**.
- Se puede ajustar **k** mediante una fase previa al uso del algoritmo kNN para clasificar.
- Normalmente se usa **validación cruzada**, p.e. 5-cv, interna sobre el conjunto de entrenamiento para determinar la bondad de un valor de k concreto.
- **Opciones:**
 - **Secuencial:** Desde $k = 1$ hasta un limite evaluar la bondad de k y elegir el mejor.
 - **Bisección:** Comenzar con $k = 1$ y doblar su valor mientras se mejore en bondad. Al empeorar, realizar búsqueda binaria en el intervalo resultante

- **Reducción de la dimensionalidad**
 - En cuanto a **número de elementos**:
 - En **tiempo de ejecución**, para cada instancia a clasificar, hay que **calcular** la **distancia a toda instancia** del conjunto de entrenamiento.
 - **Solución**: reducir el numero de instancias.
 - **Alternativas**: Condensación, Edición
 - En cuanto a **número de características**:
 - Un **número alto de características** supone agregar **complejidad** y posible **distorsión** al cálculo de las **distancias**
 - **Solución**: Selección de variables
 - **Alternativas**: Estudio de Correlación, Clustering, Árboles de decisión.

- **Condensación:** CNN retiene los puntos que hay cerca de las fronteras. Funciona iterativa e incrementalmente:
 - Comenzar con $R = \text{Conjunto Vacío}$;
 - Comprobar todos los elementos de S y encontrar una instancia e_i t.q. su vecino más cercano e_j tenga distinta clase.
 - Eliminar e_i de S y añadirlo a R .
 - Repetir hasta que no se seleccione ningún prototipo más.
 - Usar R en lugar de S para clasificar.



- **Edición: ENN elimina los puntos que hay cerca de las fronteras, considerándolos ruido. Funciona decrementalmente.**
 - Comenzar con $S = D$.
 - Fijar el valor de k , normalmente $k = 3$.
 - Eliminar de S las instancias para las que su clase no coincide con la que se predice usando sus k vecinos más cercanos.
 - Usar S en lugar de D para clasificar.

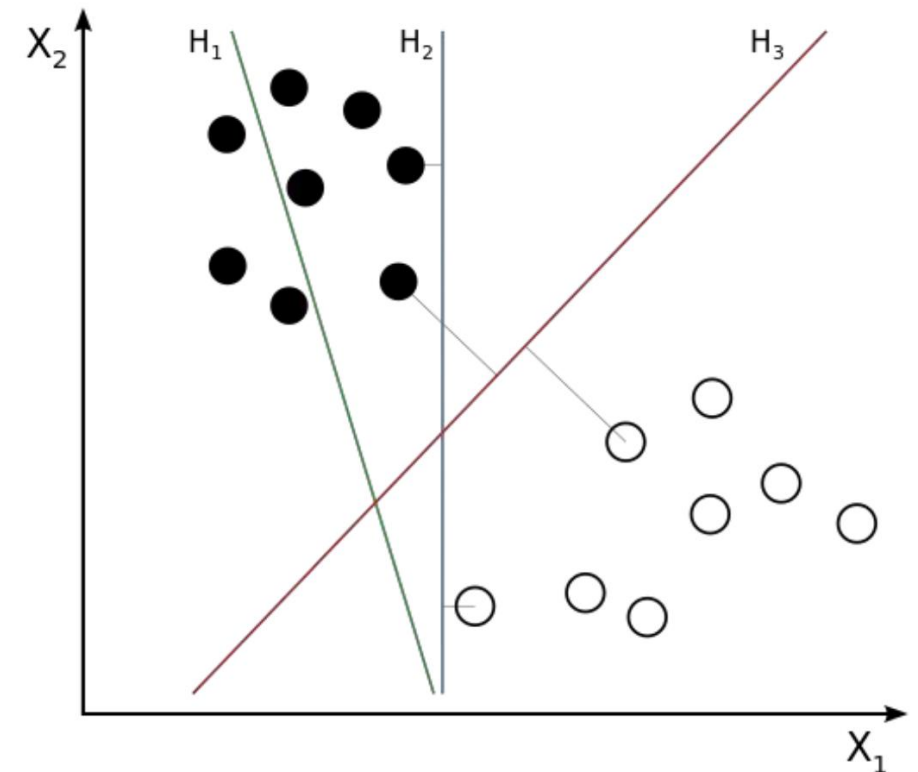
SVM

viu | **Universidad**
Internacional
de Valencia

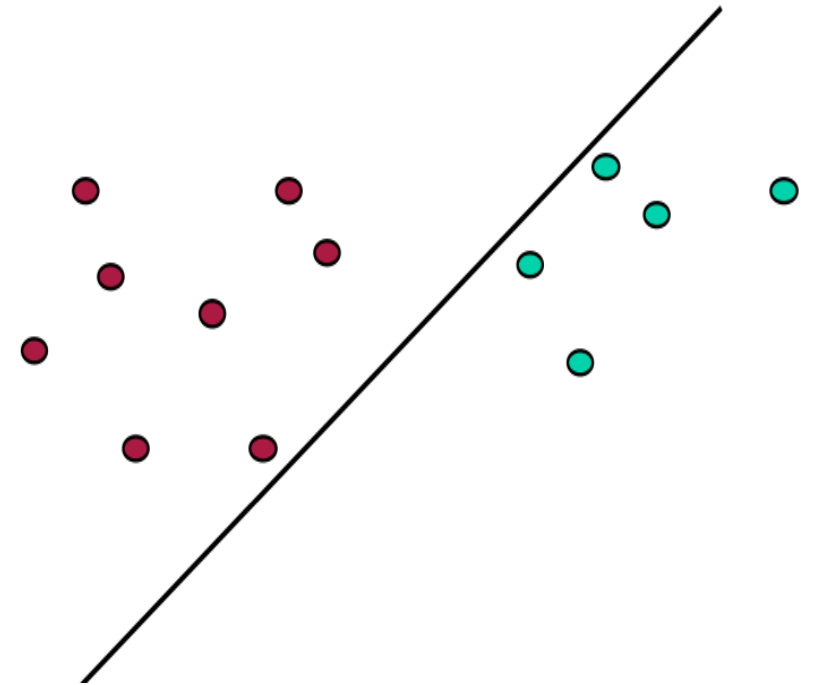


viu | **Universidad**
Internacional
de Valencia

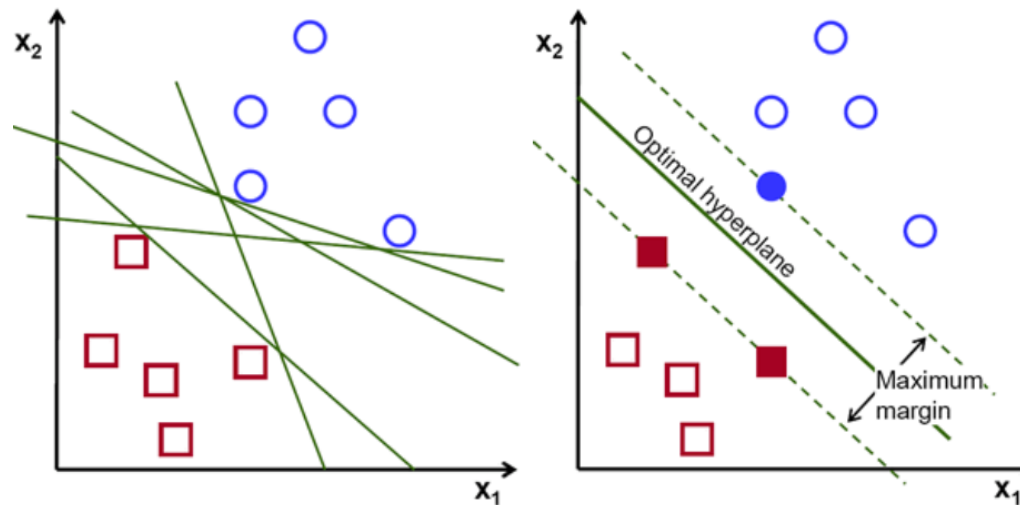
- Se definen como clasificadores lineales de **máximo margen**
- Más utilizados en **clasificación** (Support Vector Classification, SVC), aunque también pueden usarse en **regresión** (Support Vector Regression, SVR)
- Búsqueda de un **hiper-plano separador** de máximo margen de las distintas clases existentes
- Problemas con datos no linealmente separables
- **Solución:** uso de kernels



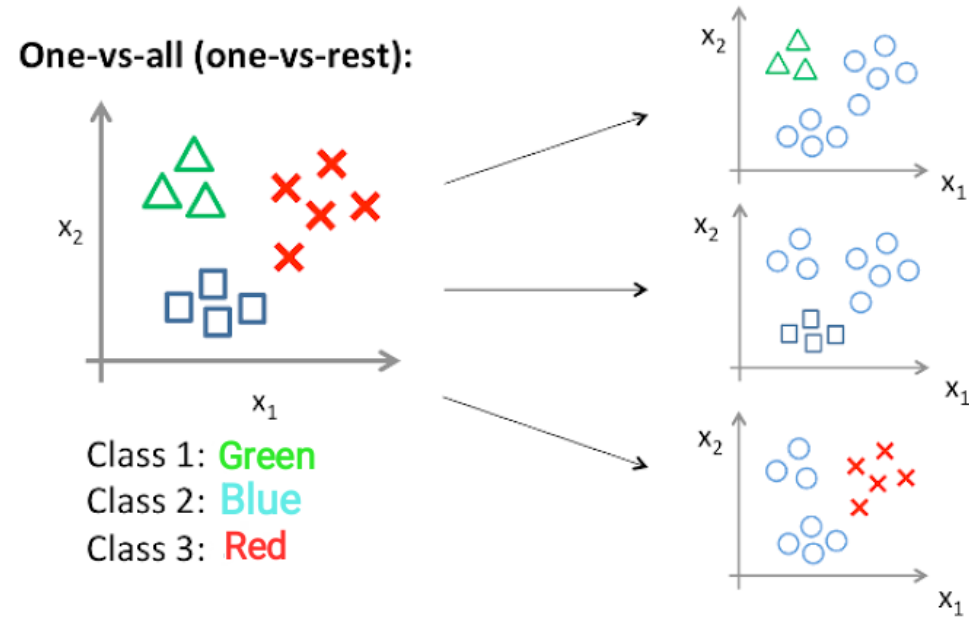
- Necesitamos encontrar a , b y c tales que:
 - para los puntos rojos
 - para los puntos verdes
- **Muchas posibles soluciones** para a , b y c
- Algunos métodos encuentran un hiper-plano de separación, pero no es el **óptimo**. P. ej.: las redes neuronales.
- ¿**Qué puntos** deberían importarnos?
 - ¿**Todos**? -> Regresión lineal, redes neuronales...
 - ¿Sólo los puntos **difíciles** que se encuentran en la **frontera** de decisión? -> **SVM**



- Los **vectores de soporte** son aquellos elementos del conjunto de datos de entrenamiento que **cambiarían** la posición del **hiper-plano** de separación si los **quitásemos** o **modificásemos** (elementos críticos).
- El problema de encontrar el hiper-plano óptimo es un **problema de optimización** que se puede **solucionar** mediante técnicas de optimización: usando los multiplicadores de **Lagrange** para formular el problema de forma que se pueda solucionar analíticamente)

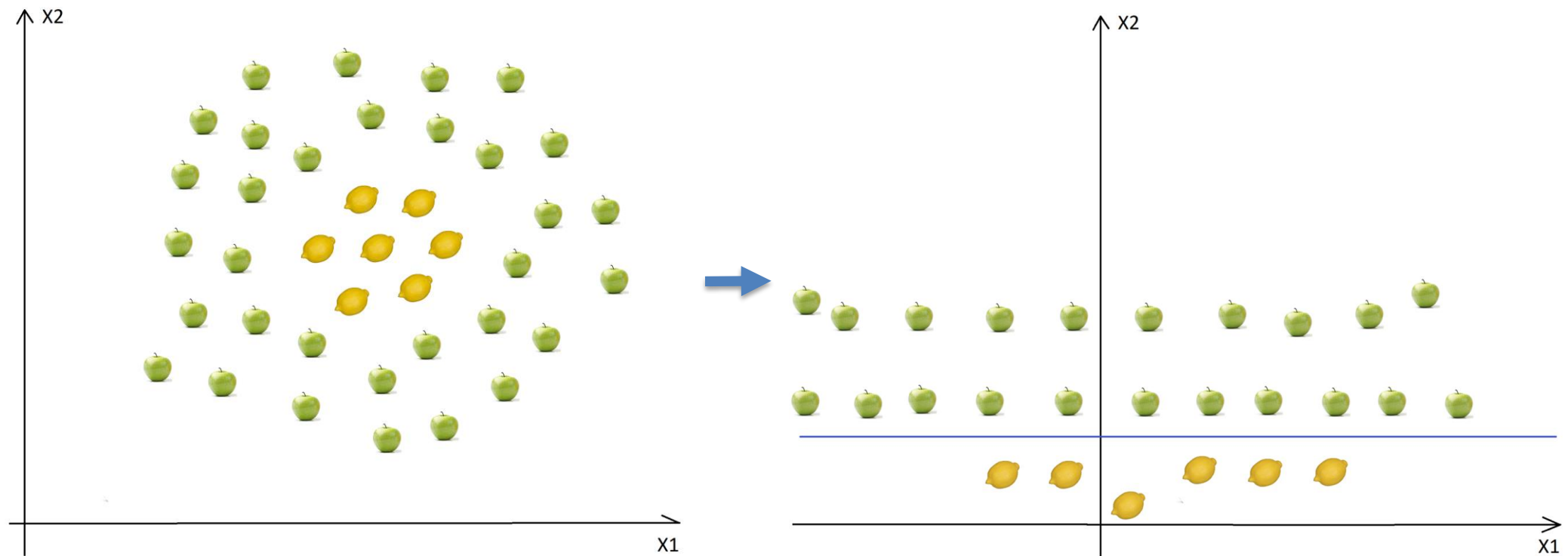


- Clases no binarias:



- En los siguientes enlaces podéis ver la formulación matemática que permite solucionar el problema:
 - <http://web.mit.edu/6.034/wwwbob/svm-notes-long-08.pdf>
 - https://ai6034.mit.edu/wiki/images/SVM_and_Boosting.pdf
 - https://xavierbourretsicotte.github.io/SVM_by_hand.html

- ¿Y si los datos no fuesen linealmente separables?
- **Solución:** utilizar una función de transformación de los datos para hacerlos linealmente separables -> “**kernel trick**”



<https://towardsdatascience.com/svm-and-kernel-svm-fed02bef1200>

VENTAJAS

- Funciona muy bien si existe un margen de **separación entre las clases**
- Funciona en espacios **altamente dimensionales**
- **Funciona** cuando el número de dimensiones es mayor que el número de instancias ($d > n$)
- Es **eficiente** en términos de memoria, ya que únicamente utiliza un **subconjunto** de los datos (los llamados vectores de soporte)

INCONVENIENTES

- **Poco eficiente** con **datasets grandes**
- **Poco eficiente** en datasets **ruidosos** (por ejemplo, con clases solapadas)
- **No** proporciona **probabilidades** de pertenencia, sino *hard-labels* (aunque hay alternativas)

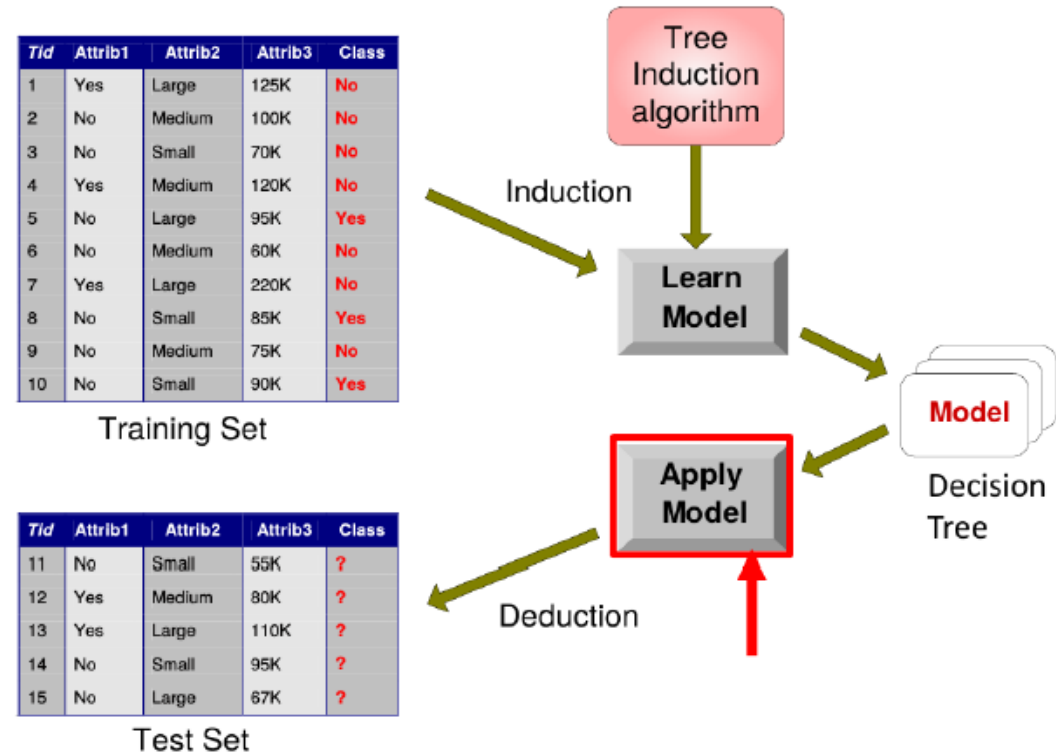
ÁRBOLES DE DECISIÓN

viu | **Universidad**
Internacional
de Valencia



viu | **Universidad**
Internacional
de Valencia

- Los **árboles de decisión** son una técnica de aprendizaje automático por **inducción** que permiten identificar conceptos (clases de objetos) a partir de las características de un conjunto de ejemplos que los representan.



- La información extraída de los mismos queda organizada jerárquicamente en forma de **árbol**, es decir, en forma de grafo dirigido que consta de nodos y arcos.
- Los nodos corresponden a una pregunta o a un test que se hace a los ejemplos.
- El árbol de decisión se construye a base de ir haciendo **preguntas** sobre características determinadas a los ejemplos y **clasificándolos** según la respuesta.

Árboles de Decisión

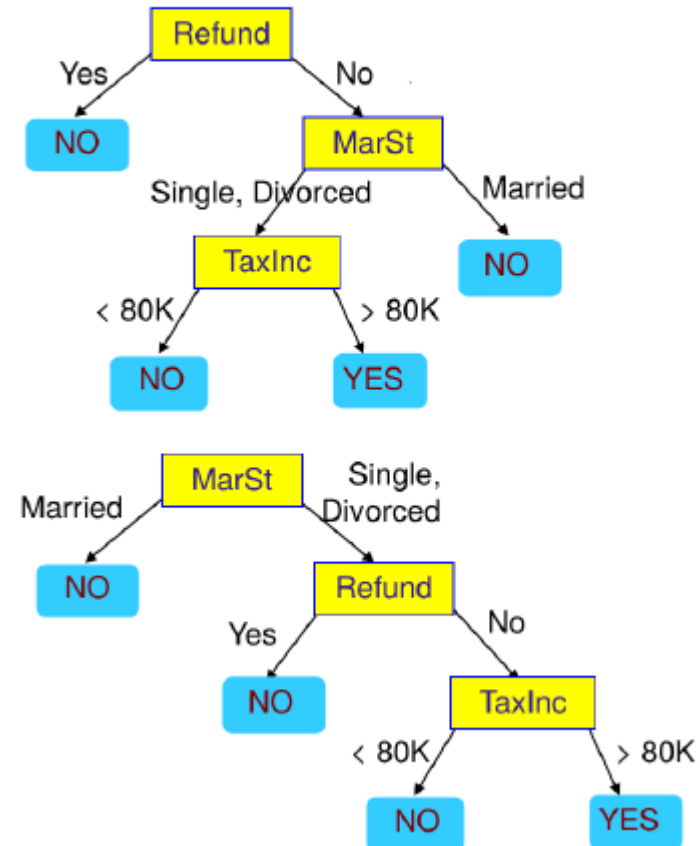
¿Qué es?

categorical
categorical
continuous
class

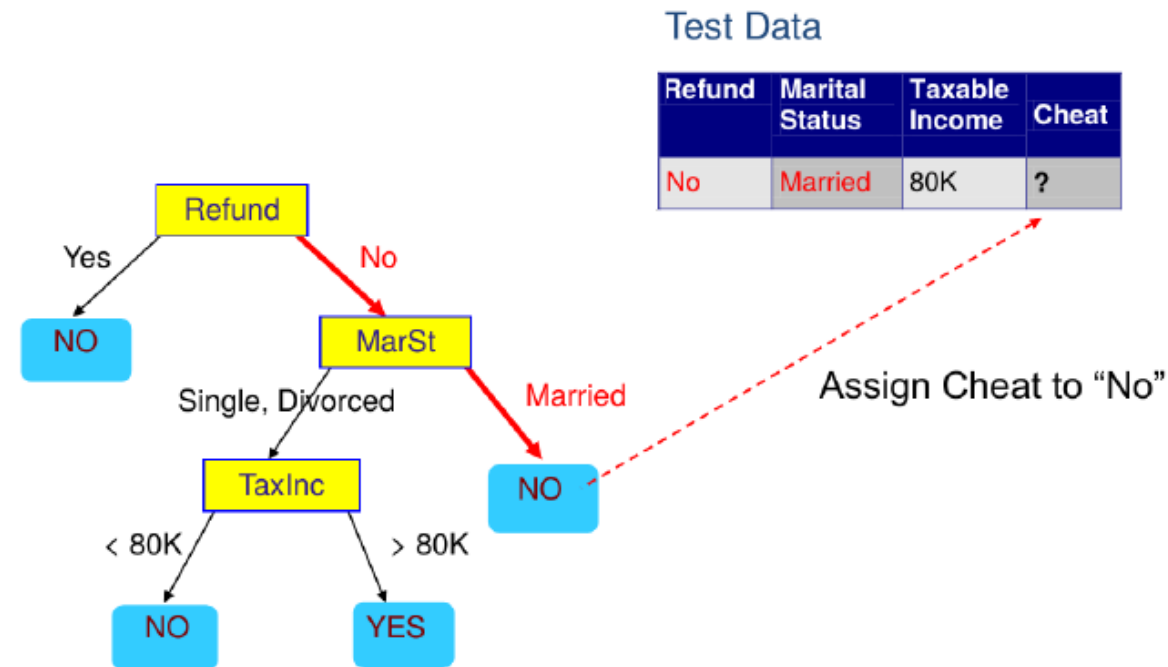
Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Training Data

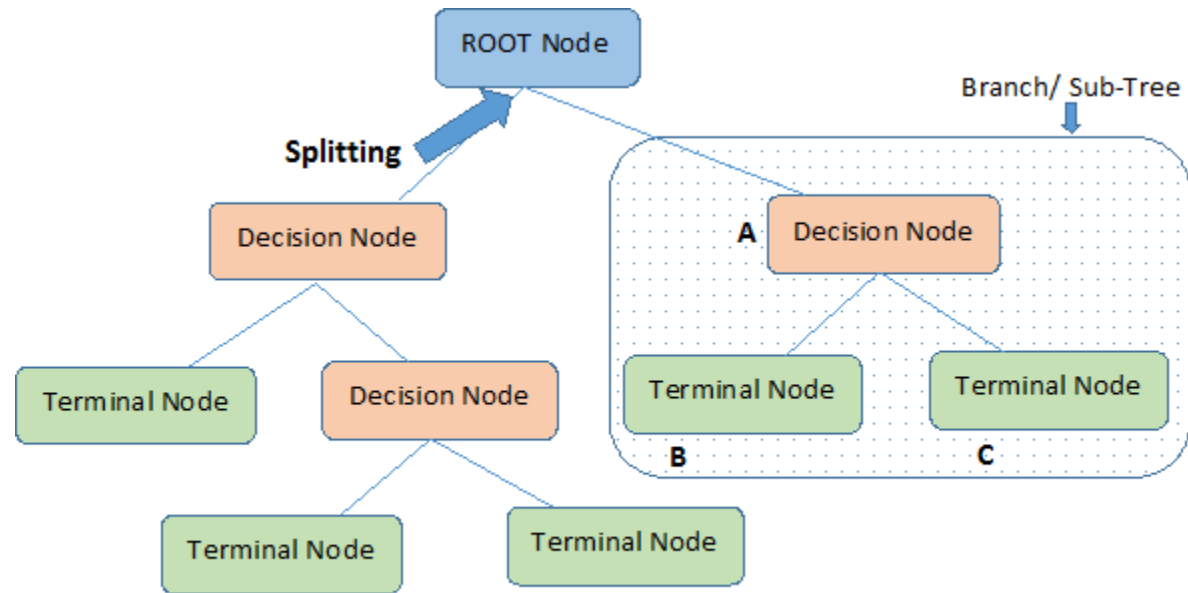
Candidate Concepts/Decision Trees



- Las diferentes opciones de clasificación (respuesta a las preguntas) son excluyentes entre sí, lo que hace que a partir de casos desconocidos y siguiendo el árbol adecuadamente, se llegue a una única conclusión o decisión a tomar.

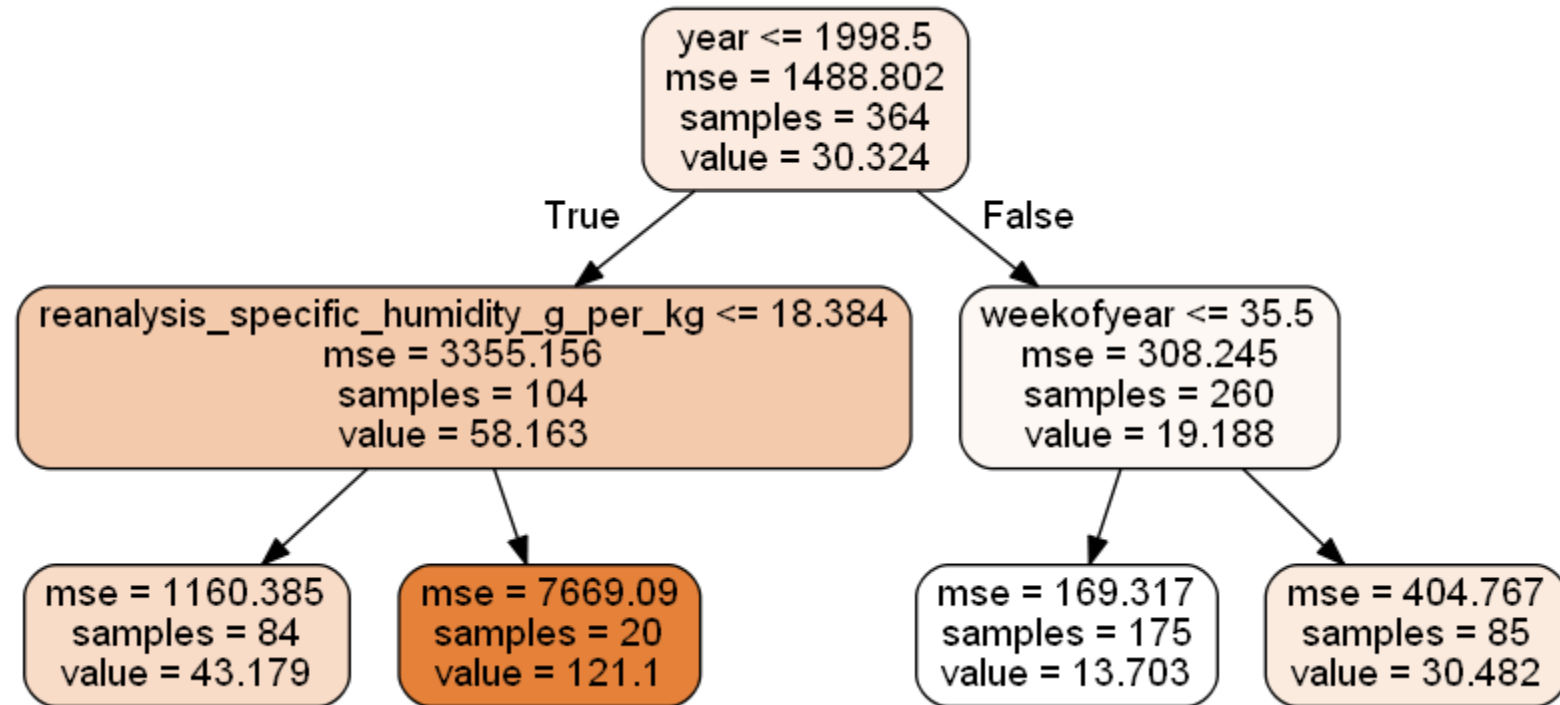


- Un árbol de decisión tiene un nodo **raíz**, nodos **intermedios** y **hojas**.
- **Nodo raíz:** Es el **principal** rasgo o **descriptor** que permite dividir el conjunto original en dos subconjuntos mejores (**aporta mayor información**)
- **Nodos Intermedios:** Cualquier nodo intermedio puede ser un **nodo raíz de un subárbol**. Esto conduce a una definición recursiva de árbol de decisión.
 - Cada nodo intermedio y la raíz tienen asociados separadores que formulan una pregunta o realizan un test acerca de la existencia o no de una característica en cada caso ejemplo. Esto permite clasificar los ejemplos y determinar cuáles serían los nodos sucesores.
- **Hojas:** Una hoja en el árbol **corresponde a un conjunto de ejemplos** que representan una sola clase. La clase de la hoja se asigna por el **criterio de a la que pertenezcan la mayoría de los ejemplos** en ella. Las hojas del árbol de decisión representan los conceptos extraídos de manera automática



Note:- A is parent node of B and C.

<https://www.analyticsvidhya.com/blog/2016/04/complete-tutorial-tree-based-modeling-scratch-in-python/>



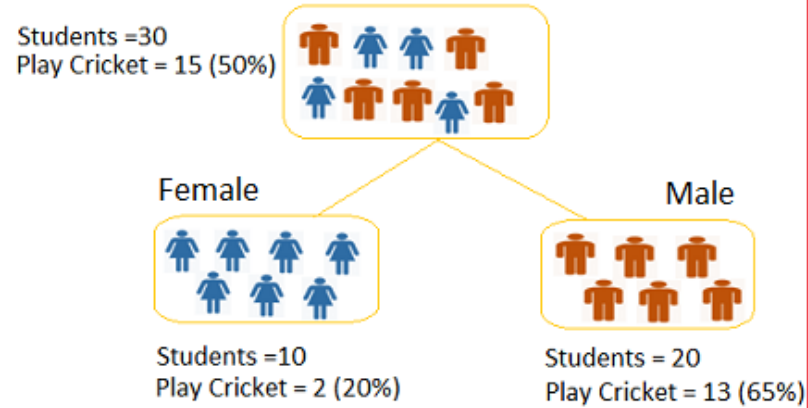
- El algoritmo general para todo método basado en árboles de decisión sería el siguiente:
 - Se calcula la calidad (impureza, información, etc.) del nodo (conjunto de ejemplos).
 - Si un conjunto de ejemplos en un nodo no tiene suficiente calidad, se calcula **el mejor atributo separador**.
 - Una vez que se obtiene, se añade a la base de reglas, y se utiliza para dividir el conjunto de ejemplos en al menos dos conjuntos nuevos de mayor calidad.
 - Seguir separando hasta que se cumpla el **criterio de parada**.

- El rasgo o **descriptor** que **seleccionar** debe de cumplir el **objetivo** de que **su posición** en algún punto del árbol **genere** un **subárbol** tan **simple** como sea posible y dé una concreta **clasificación**.
- De esta forma, cuando se construye un árbol de decisión, es necesario tener un medio para determinar tanto los **atributos importantes** requeridos para la clasificación, como el **orden** de uso de esos atributos importantes.
- Es por ello por lo que es clave la selección del **criterio de selección** de separadores lo cual dará lugar a diferentes algoritmos de construcción de árboles de decisión.

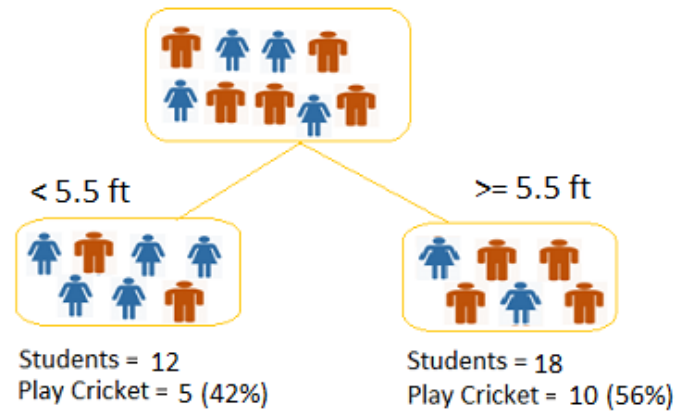
Árboles de Decisión

¿Cómo se construye?

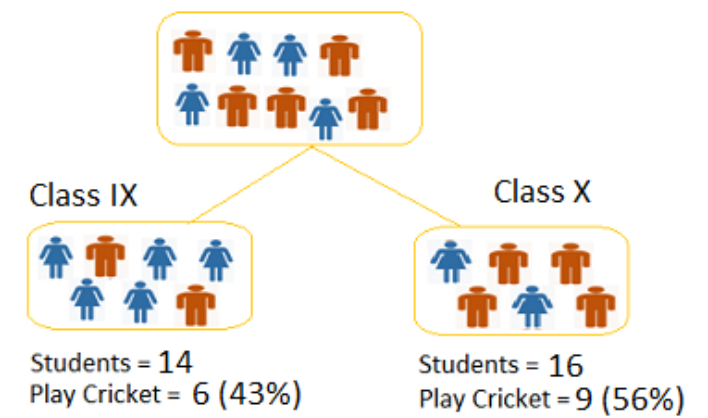
Split on Gender



Split on Height



Split on Class



<https://www.analyticsvidhya.com/blog/2016/04/complete-tutorial-tree-based-modeling-scratch-in-python/>

- **Índice de Entropía:** se calcula el índice de entropía para cada una de las variables (el “desorden”). Si la variable es categórica, se obtiene sumando el índice de entropía de cada una de sus clases. Si es numérica, previamente se obtiene uno o varios puntos de corte por métodos iterativos. Se elegirá aquella variable que tenga menor índice de entropía (aquella cuya proporción de casos positivos esté más alejada de 0,5).
- **Chi-cuadrado:** mide el **grado de asociación entre dos variables**. Se calcula comparando la tabla de contingencia dada por el cruce de la variable objetivo versus cada una de las variables explicativas con una tabla donde no hubiera asociación. Se elegirá la variable con mayor valor del estadístico chi-cuadrado.
- **Índice de Gini:** mide la **probabilidad de no sacar dos registros con el mismo valor para la variable objetivo dentro del mismo nodo**. Cuando menor es el índice de Gini mayor es la pureza del corte. Por lo tanto, el corte propuesto en primer lugar será el de aquella variable que tenga menor valor del índice de Gini.

$$\text{Entropy}(t) = - \sum_{i=0}^{c-1} p(i|t) \log_2 p(i|t),$$

$$\text{Gini}(t) = 1 - \sum_{i=0}^{c-1} [p(i|t)]^2,$$

$$\text{Classification error}(t) = 1 - \max_i [p(i|t)],$$

p refers to the fraction of records that belong to one of the two classes

Node N_1	Count
Class=0	0
Class=1	6

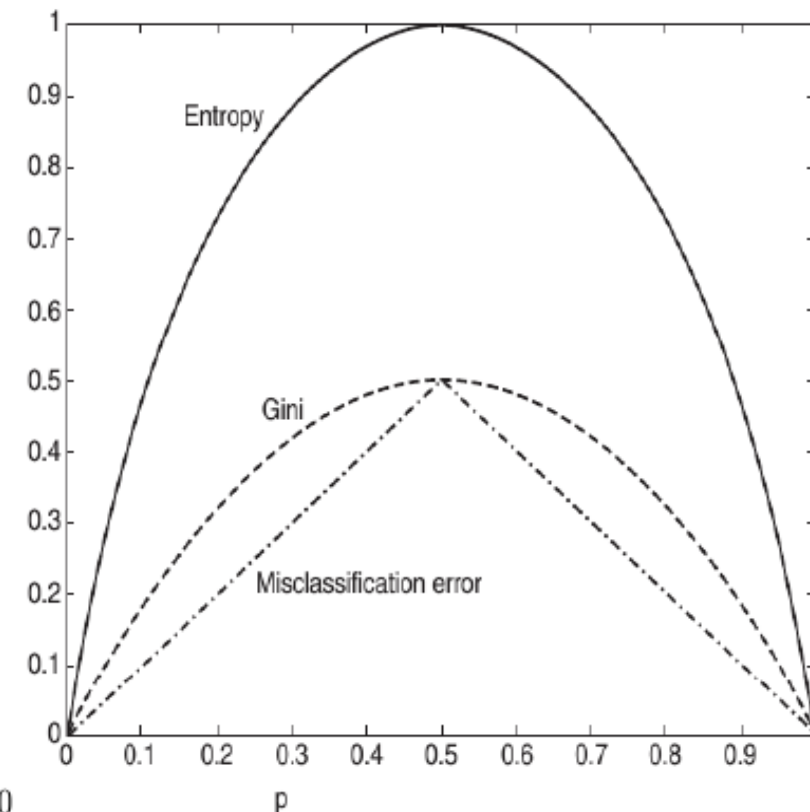
$$\begin{aligned} \text{Gini} &= 1 - (0/6)^2 - (6/6)^2 = 0 \\ \text{Entropy} &= -(0/6) \log_2(0/6) - (6/6) \log_2(6/6) = 0 \\ \text{Error} &= 1 - \max[0/6, 6/6] = 0 \end{aligned}$$

Node N_2	Count
Class=0	1
Class=1	5

$$\begin{aligned} \text{Gini} &= 1 - (1/6)^2 - (5/6)^2 = 0.278 \\ \text{Entropy} &= -(1/6) \log_2(1/6) - (5/6) \log_2(5/6) = 0.650 \\ \text{Error} &= 1 - \max[1/6, 5/6] = 0.167 \end{aligned}$$

Node N_3	Count
Class=0	3
Class=1	3

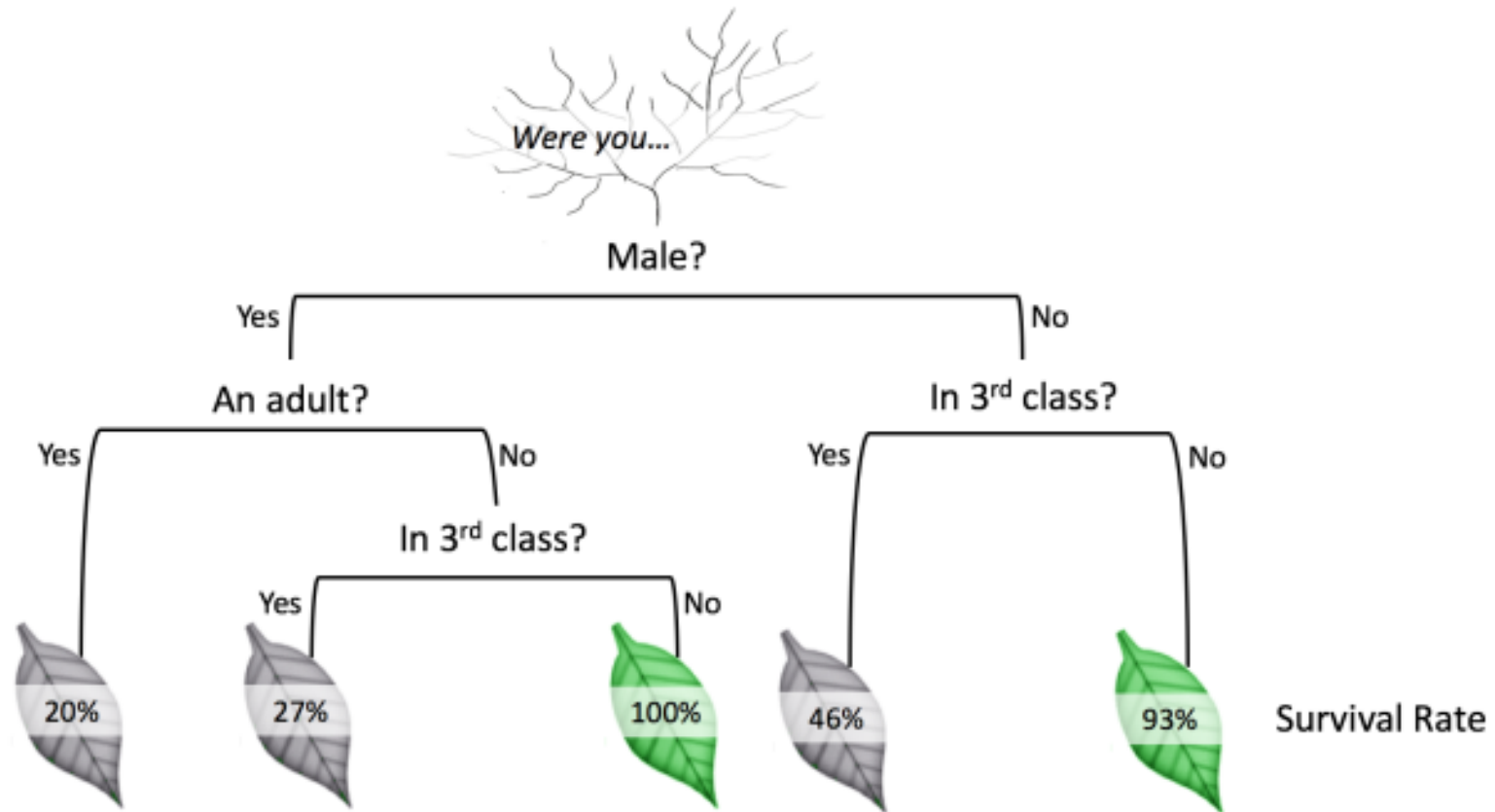
$$\begin{aligned} \text{Gini} &= 1 - (3/6)^2 - (3/6)^2 = 0.5 \\ \text{Entropy} &= -(3/6) \log_2(3/6) - (3/6) \log_2(3/6) = 1 \\ \text{Error} &= 1 - \max[3/6, 3/6] = 0.5 \end{aligned}$$



- **Máximo número de ramas por nodo (Maximum Branch):** número de ramas máximo en que puede dividirse un nodo.
- **Número mínimo de observaciones por nodo final (Leaf Size):** número mínimo de observaciones que tiene que tener un nodo final para que se construya la regla.
- **Número mínimo de observaciones para dividir un nodo (Split Size):** número mínimo de observaciones que tiene que tener un nodo para que se pueda cortar por la variable seleccionada.
- **Variables discriminantes (Discriminant Variables):** no encontrar ninguna variable que sea lo suficientemente discriminante en el nodo es motivo de parada.

Árboles de Decisión

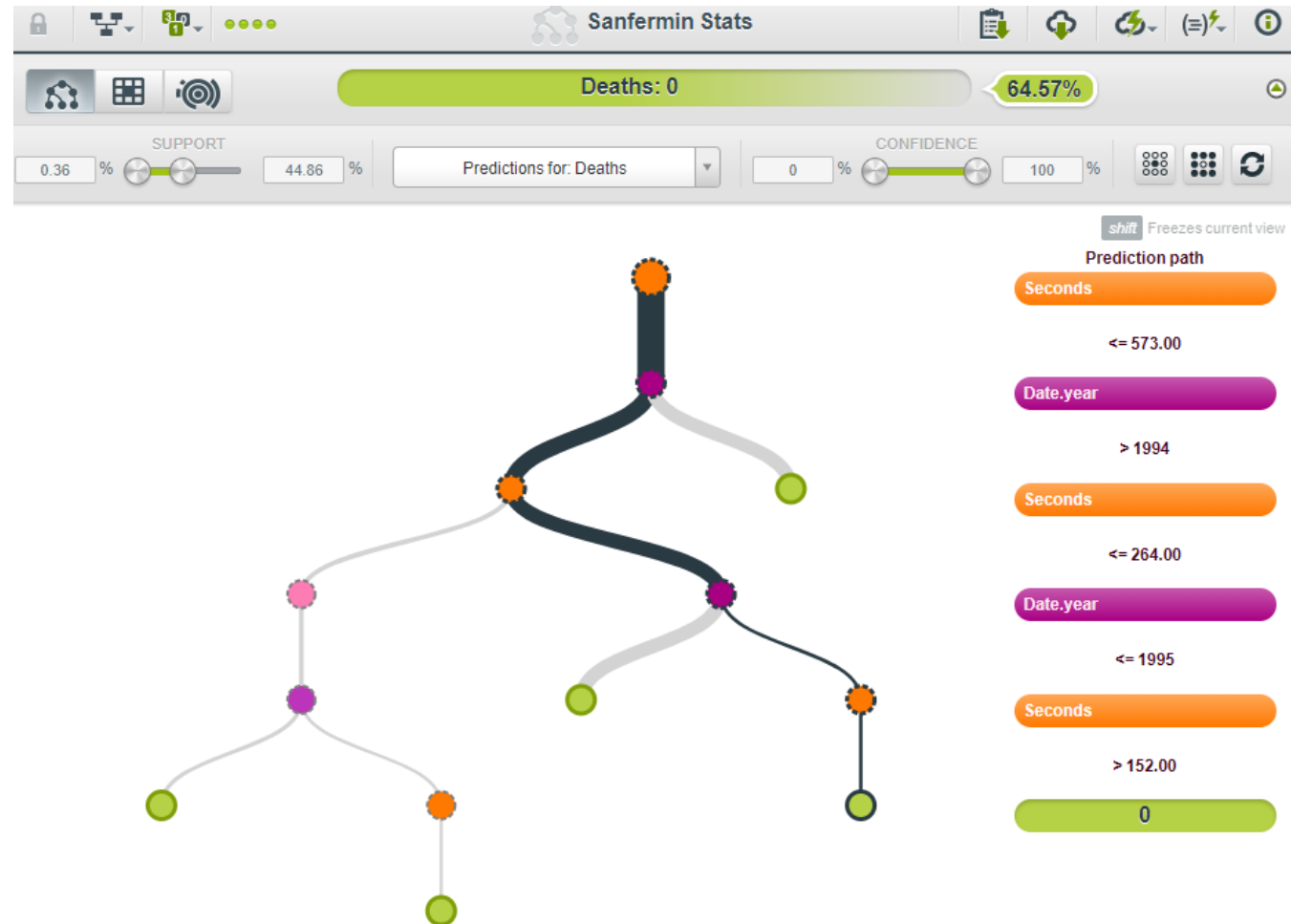
Ejemplo



<https://algorithmeans.com/2016/07/27/decision-trees-tutorial/>

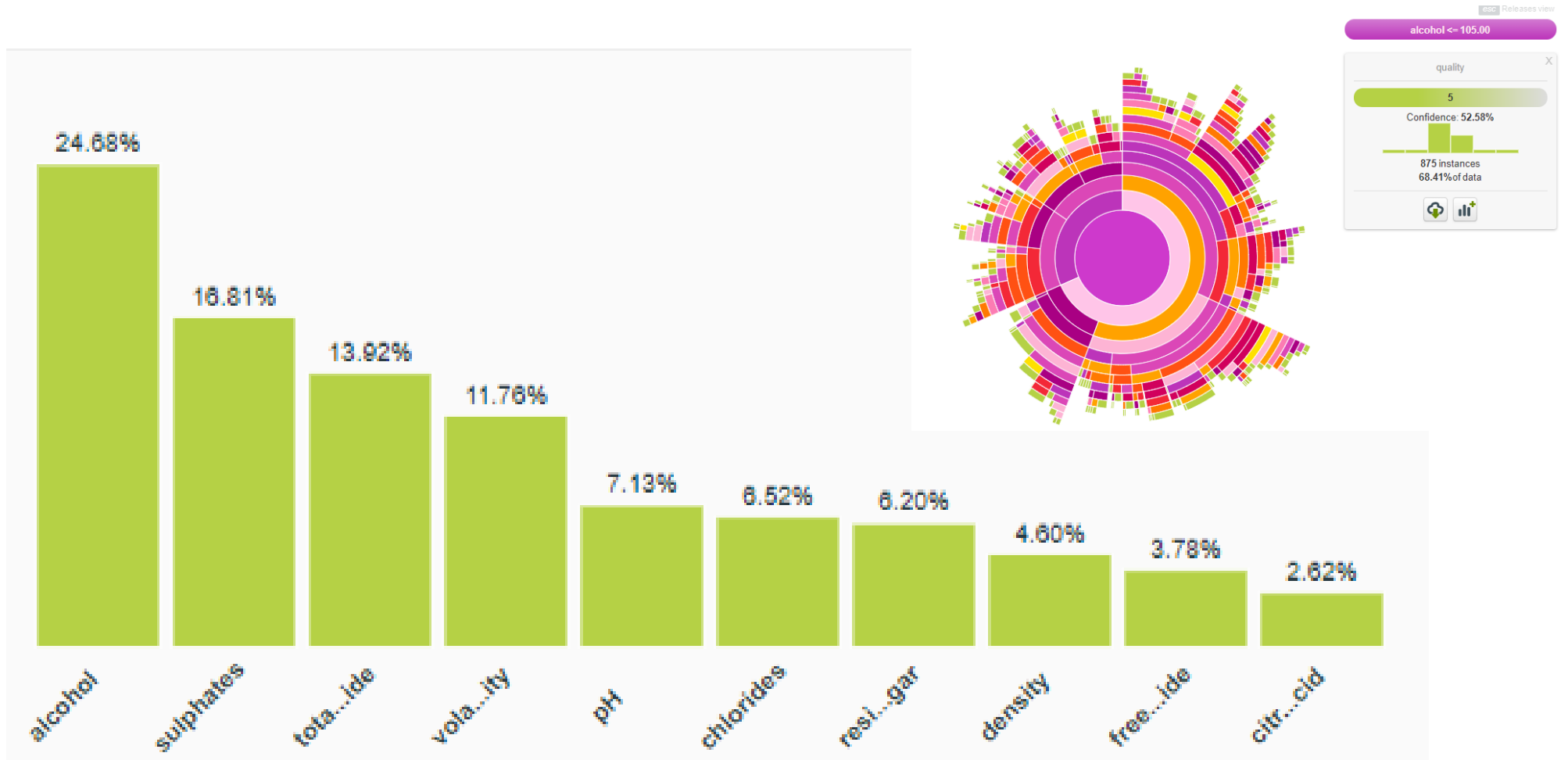
Árboles de Decisión

Ejemplo



Árboles de Decisión

Ejemplo



- **Podar** el árbol consiste en **reducirlo**, haciéndolo más **sencillo dejando** sólo los nodos más **importantes** y a su vez **eliminando** los **redundantes**.
- Para ello se sigue habitualmente el siguiente procedimiento:
 - Antes de llamar **recursivamente** al algoritmo para cada Nodo:
 - Se calcula la **tasa de ramificación** del **conjunto actual**.
 - Se calcula la **tasa de ramificación** de los **subconjuntos**.
 - Si la tasa de ramificación del Subconjunto 1 es **menor** que la del conjunto, se aplica el algoritmo sobre él, si no se detiene la expansión.
 - Si la tasa de ramificación del Subconjunto 2 es **menor** que la del conjunto, se aplica el algoritmo sobre él, si no se detiene la expansión.
 - ...

	Splitting Criteria	Attribute type	Missing values	Pruning Strategy	Outlier Detection
ID3	Information Gain	Handles only Categorical value	Do not handle missing values.	No pruning is done	Susceptible to outliers
CART	Towing Criteria	Handles both Categorical & Numeric value	Handle missing values.	Cost-Complexity pruning is used	Can handle Outliers
C4.5	Gain Ratio	Handles both Categorical & Numeric value	Handle missing values.	Error Based pruning is used	Susceptible to outliers

<https://www.quora.com/What-are-the-differences-between-ID3-C4-5-and-CART>

VENTAJAS

- **No son costosos** de construir
- Extremadamente **rápidos** al tratar nuevos ejemplos
- Si el árbol es pequeño, el **modelo es muy interpretable**
- El **rendimiento** es similar a otros modelos más complejos.

INCONVENIENTES

- Los árboles grandes son **difíciles de entender** y tienden al **sobreaprendizaje**.
- Algunos resultados **dependen** del algoritmo elegido.

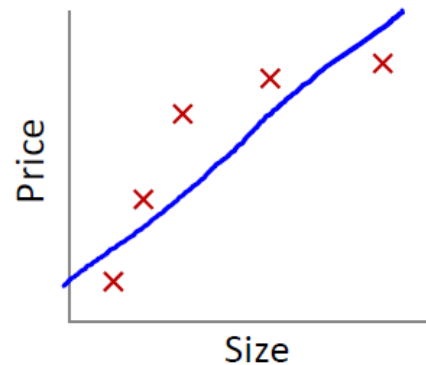
CONJUNTOS DE MODELOS

viu | **Universidad**
Internacional
de Valencia

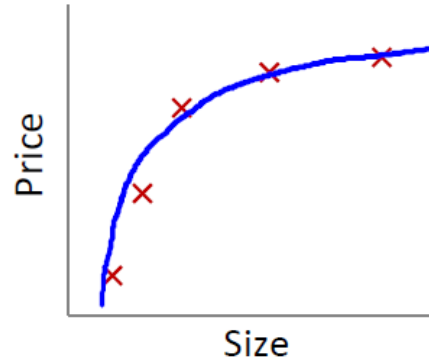


viu | **Universidad**
Internacional
de Valencia

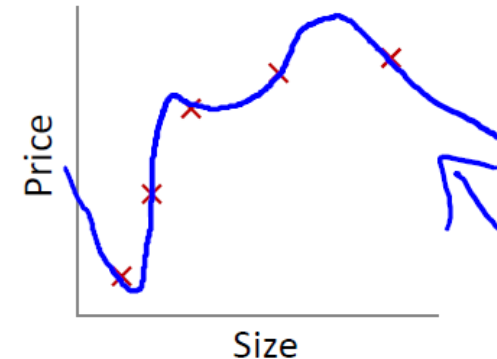
- Uno de los inconvenientes de los modelos individuales es que puede sobreaprender, es decir, sobreajustar sus datos, por lo que su rendimiento en su entrenamiento los datos son muy buenos, pero no se generalizan bien a los datos nuevos, lo que hace que los modelos individuales de árboles de decisión sean más peores modelos.
- Para solucionar este problema aparecen los conjuntos de modelos.



$\rightarrow \theta_0 + \theta_1 x$
"Underfit" "High bias"

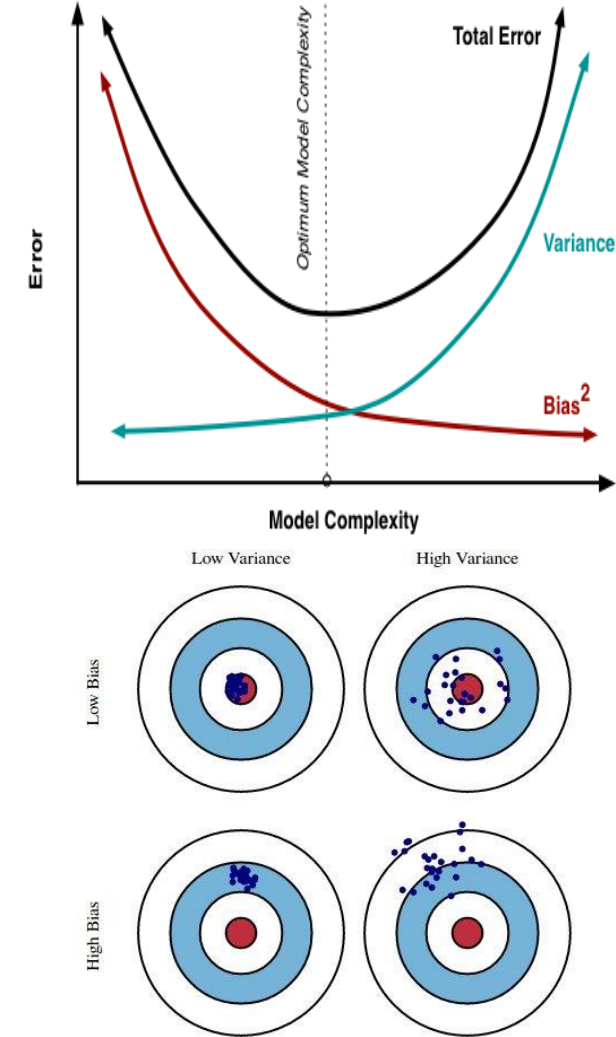


$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2$
"Just right"



$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$
"Overfit" "High variance"

- **Bias (sesgo)**
 - Error introducido por aproximar una función compleja con un modelo excesivamente simple.
 - Diferencia entre lo predicho y lo real
 - **Under-fitting**
- **Variance (varianza)**
 - Capacidad del modelo para adaptarse si se estimara con un conjunto de entrenamiento diferente
 - Si un modelo tiene una alta varianza, cualquier mínimo cambio le afectaría.
 - **Over-fitting**



- Un conjunto (**ensemble**) es una **colección** de árboles de decisión múltiples que se combinan para crear un modelo **más sólido** con un **mejor rendimiento** predictivo.
- Un conjunto de modelos basados en muestras de los datos puede convertirse en un buen predictor al **promediar** los errores de cada modelo individual.
- Los conjuntos son **menos sensibles** a los valores atípicos en sus datos de entrenamiento, **evitan** el riesgo de **sobreajuste** y **generalizar mejor** cuando se los aplica a nuevos datos.
- Dependiendo de la naturaleza de sus datos y los valores específicos para los parámetros del conjunto, **puede aumentar significativamente el rendimiento** sobre el uso de un solo modelo además de permitir la paralelización para la construcción y explotación del mismo.

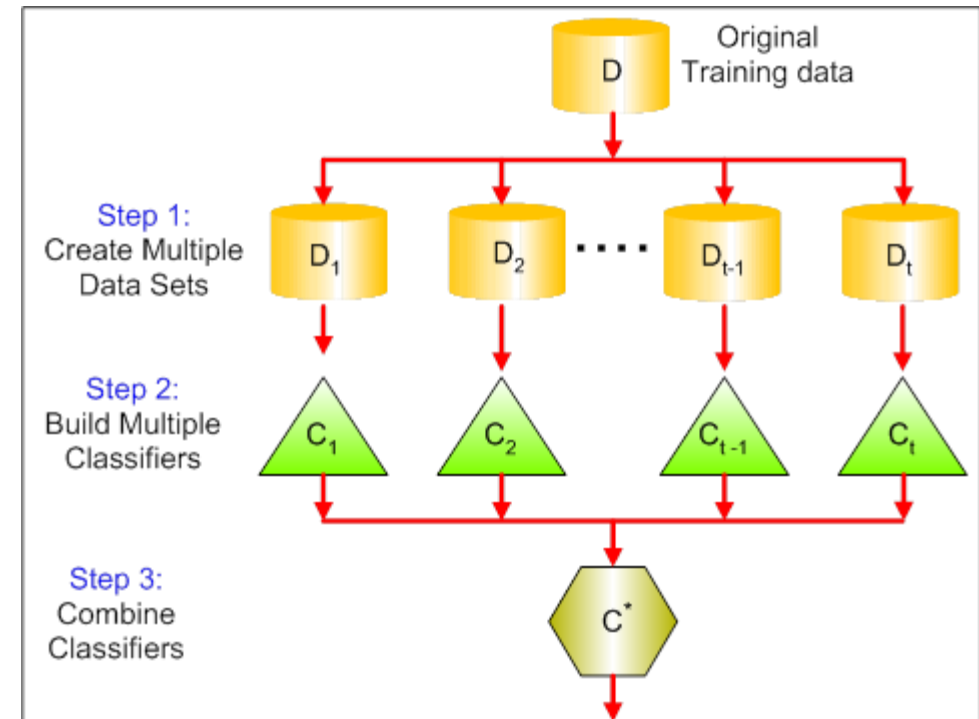
- La idea básica es **re-muestrear los datos** y calcular las predicciones sobre el conjunto de datos re-muestreados.
- Al **promediar** varios **modelos** conjuntamente se obtiene un **mejor ajuste** debido a que se mitigan tanto los modelos con **sesgo** como los modelos con alta **varianza**.
- Si el algoritmo predice datos **categoricos**, entonces el **voto** de la **mayoría** dará la clase dominante o con mejor predicción.
- Si se está realizando predicción sobre datos **numéricos**, entonces se realizará la **media** sobre las predicciones.

Bagging

Boostin
g

Random
Forests

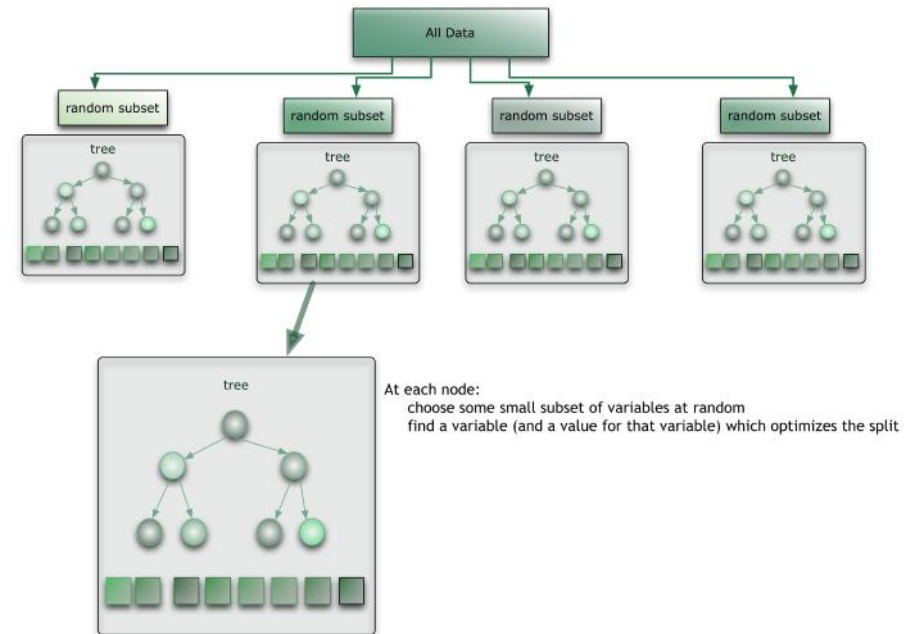
- **Bagging** (Bootstrap Aggregating): este algoritmo crea **cada árbol individual a partir del aprendizaje sobre una muestra aleatoria de los elementos** del conjunto de datos de entrenamiento.
- Por defecto las muestras son tomadas utilizando un **ratio del 100%**, reduciéndose si el conjunto de datos es muy grande, con reemplazo.
- El tamaño es el **mismo** que el conjunto de entrenamiento original **pero no su composición**.



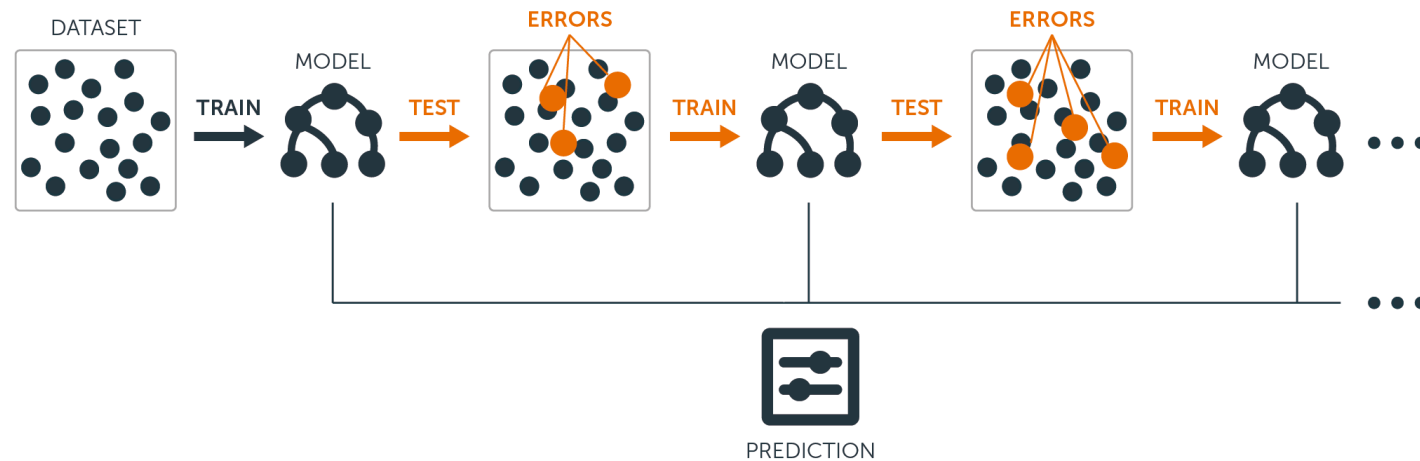
Bagging Example (Opitz, 1999)

Original	1	2	3	4	5	6	7	8
Training set 1	2	7	8	3	7	6	3	1
Training set 2	7	8	5	6	4	2	7	1
Training set 3	3	6	2	7	5	6	2	2
Training set 4	4	5	1	4	6	4	3	8

- **Random Forests:** sigue una estrategia similar al Bagging pero añadiendo un elemento adicional de aleatoriedad eligiendo para cada árbol un **subconjunto de las características del conjunto de entrenamiento**
- **Random subspaces:** cada miembro es entrenado con todos los ejemplos, pero con un **subconjunto de los atributos**.
- Parametrización
 - **Número** de arboles: SQRT
 - **Profundidad** del árbol indefinida



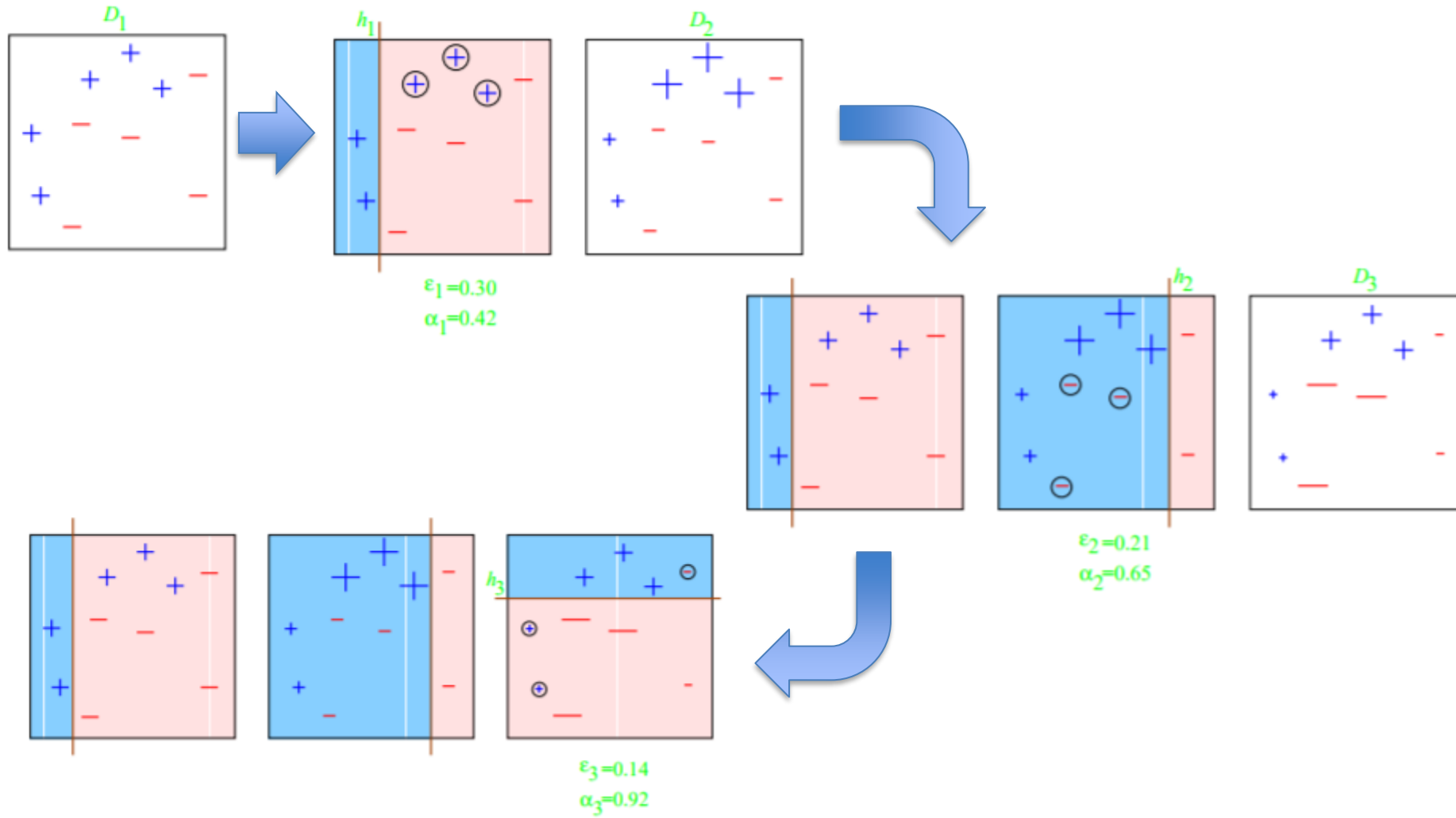
- **Boosting** (Adaboost, Gradient Boosted Trees): en este caso el algoritmo **construye secuencialmente un conjunto de árboles** de decisión que supone **modelos débiles** (weak learners, un poco mejores que aleatorios).
- En cada iteración del algoritmo, cada modelo individual intenta **corregir los errores cometidos** en la iteración previa mediante la optimización de una función de pérdida

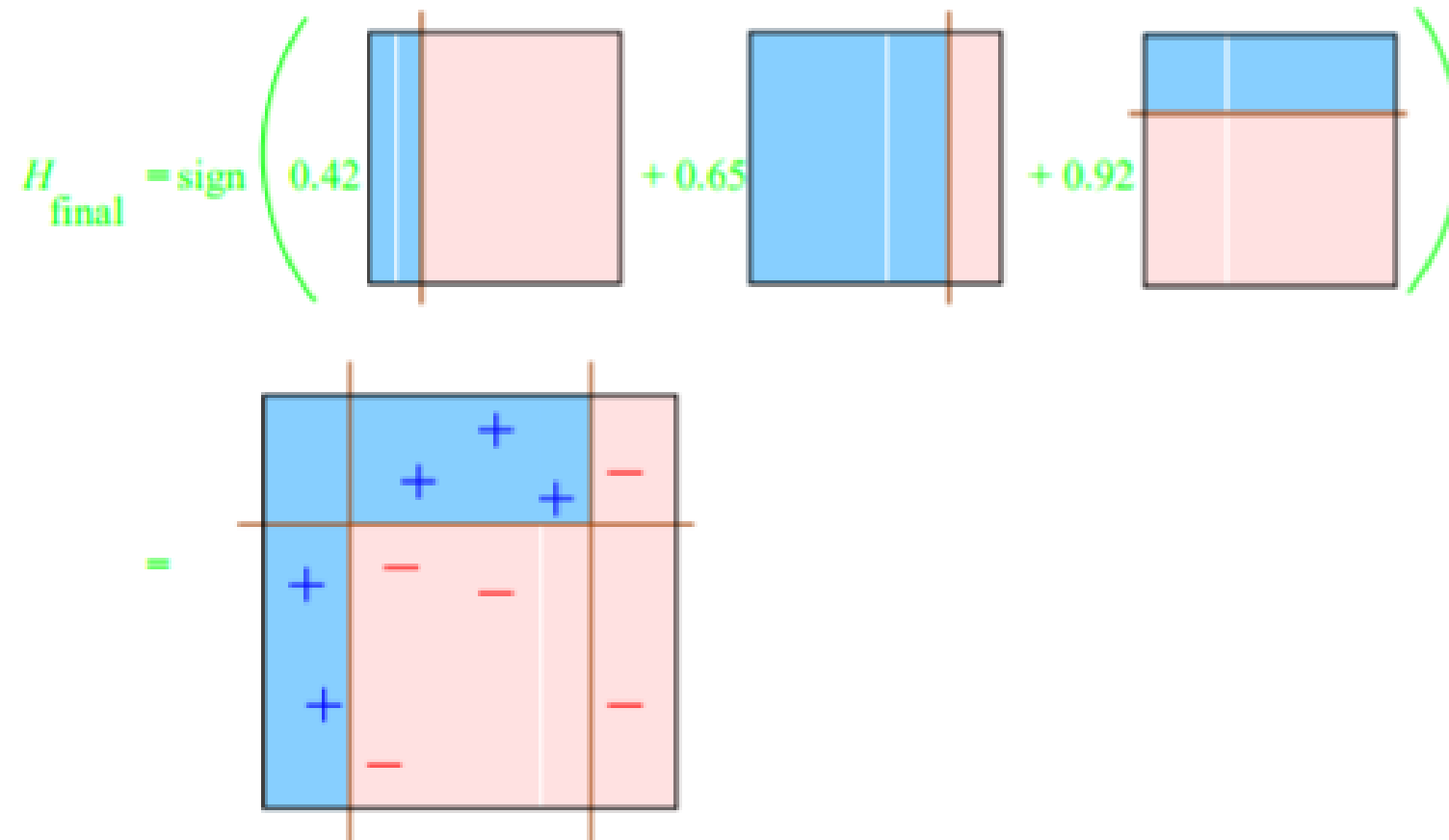


1. Inicialmente a **todos los datos** del conjunto de entrenamiento se les asigna un **peso idéntico**, $w_i = 1/n$, donde n es el tamaño del conjunto de datos.
2. Se **entrena el modelo** usando el set de entrenamiento.
3. Se **calcula error** del modelo en el set de entrenamiento, se cuentan cuántos objetos han sido **mal clasificados** y se identifican cuáles son.
4. Se **incrementan pesos** en los casos de entrenamiento que el **modelo** calcula **erróneamente**.
5. Se **entrena** un nuevo modelo usando el conjunto de **pesos** modificados.
6. Volver al punto 3 (repetir hasta el número de iteraciones fijadas inicialmente).
7. Modelo final: **votación** ponderada por los pesos de todos los modelos

Conjuntos de Modelos

Adaboost





- Elements: people in Mexico D.F
- Features: Age, Gender, Education, Residence, Industry
- Class: Salary Band:
 - Band 1 : Below 40,000
 - Band 2: 40,000 150,000
 - Band 3: More than 150,000
- Random forest:
 - 10k observaciones
 - 1 variable
 - 5 árboles (decisores)

Conjuntos de Modelos

Ejemplo

	Salary Band	1	2	3
Age	Below 18	90%	10%	0%
	19-27	85%	14%	1%
	28-40	70%	23%	7%
	40-55	60%	35%	5%
	More than 55	70%	25%	5%

	Salary Band	1	2	3
Education	<=High School	85%	10%	5%
	Diploma	80%	14%	6%
	Bachelors	77%	23%	0%
	Master	62%	35%	3%

	Salary Band	1	2	3
Gender	Male	70%	27%	3%
	Female	75%	24%	1%

	Salary Band	1	2	3
Residence	Metro	70%	20%	10%
	Non-Metro	65%	20%	15%































	Salary Band	1	2	3
Industry	Finance	65%	30%	5%
	Manufacturing	60%	35%	5%
	Others	75%	20%	5%

<http://www.analyticsvidhya.com/blog/2014/06/introduction-random-forest-simplified/>



- Proceso de Inferencia: Media de Probabilidades: votación
 - EL resultado final será la media de la probabilidad que arroja cada uno de los 5 CARTs.
- Example: 1. Age : 35 years , 2, Gender : Male , 3. Highest Educational Qualification : Diploma holder, 4. Industry : Manufacturing, 5. Residence : Metro

CART	Band	1	2	3
Age	28-40	70%	23%	7%
Gender	Male	70%	27%	3%
Education	Diploma	80%	14%	6%
Industry	Manufacturing	60%	35%	5%
Residence	Metro	70%	20%	10%
Final probability		70%	24%	6%

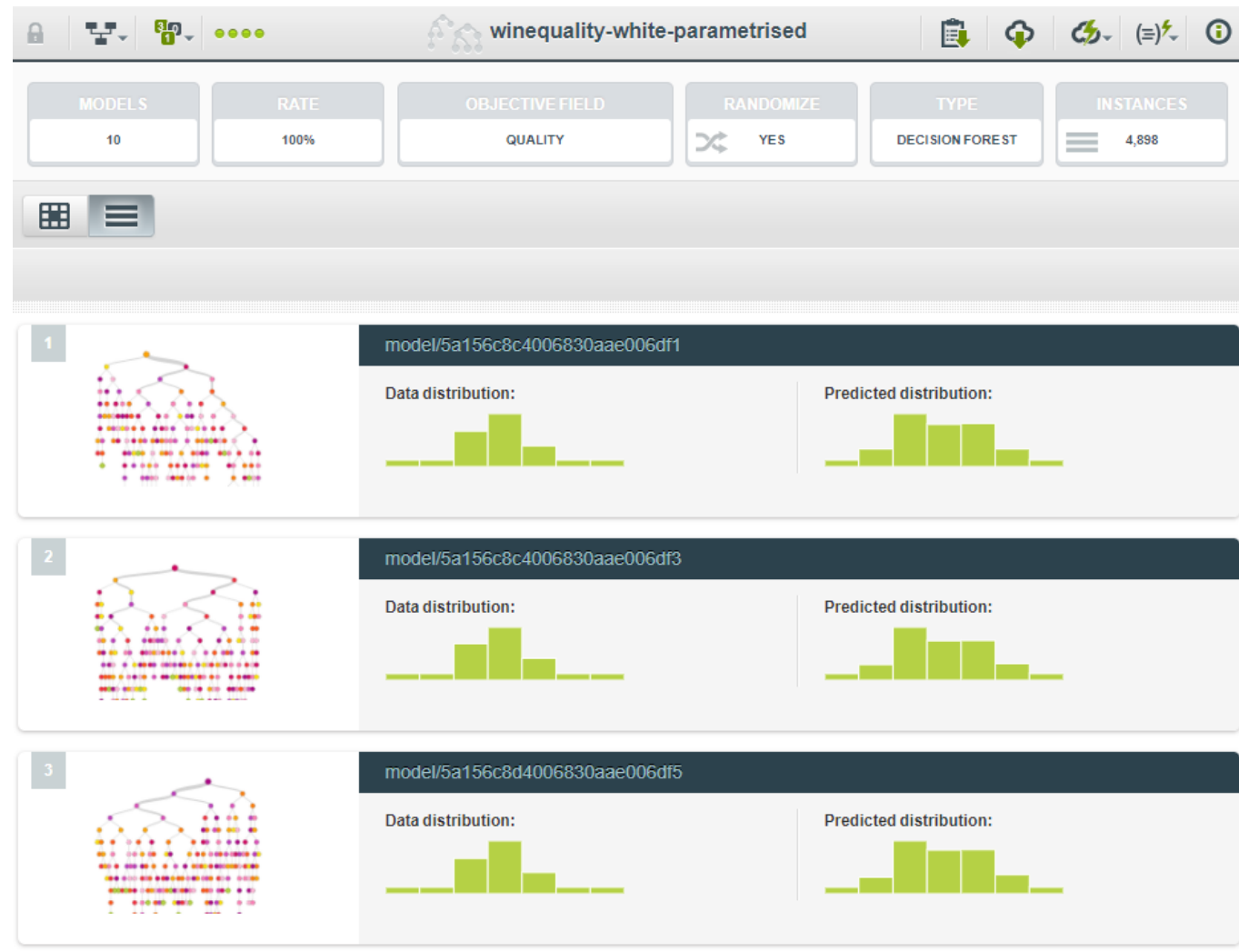
Model 1	         	7/10 correct
Model 2	         	7/10 correct
Model 3	         	6/10 correct
Ensemble Model (Majority Voting)		

For more tutorials: algobears.com

<https://algobears.com/2016/08/25/random-forest-tutorial/>

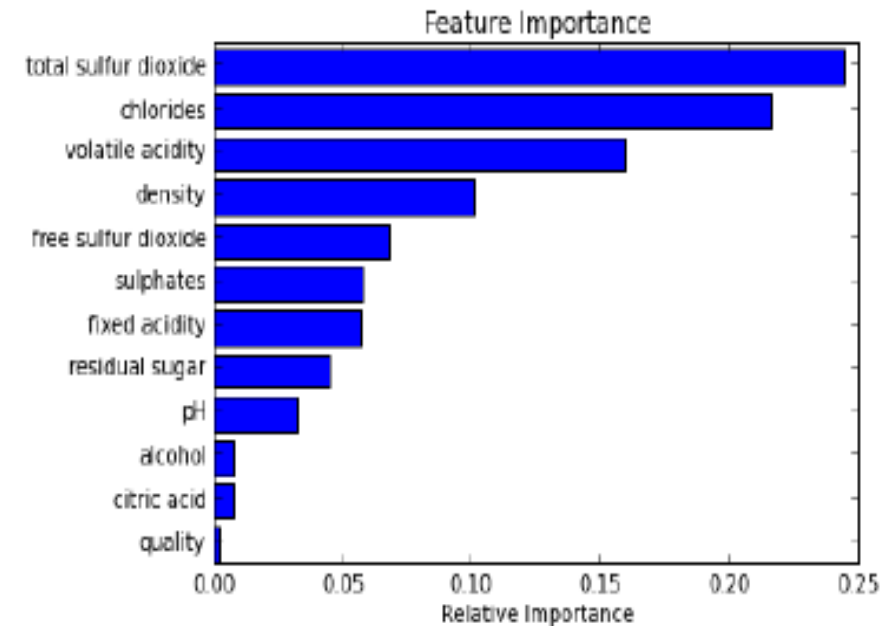
Conjuntos de Modelos

BigML

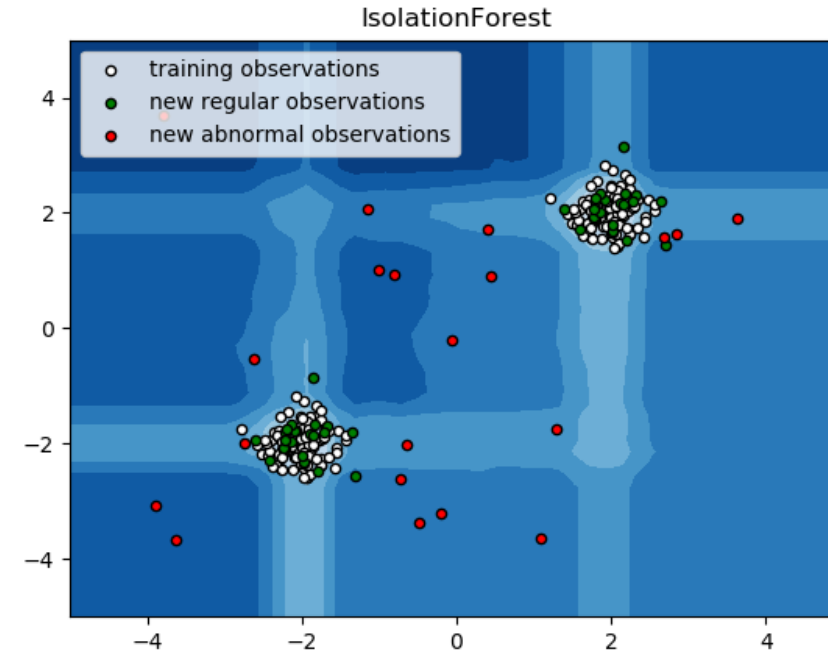


- Para medir el error de random forest se suele utilizar la técnica **denominada out-the-bag error**. Para cada árbol se utiliza el conjunto de objetos no seleccionados por su muestra bootstrap de entrenamiento para ser clasificados con dicho árbol. Promediando sobre el conjunto de árboles de random forest se puede estimar el error del algoritmo.
- Por otro lado, random forest puede ser **paralelizado** eficazmente puesto que cada árbol puede construirse de manera independiente a los otros árboles.
- Un incremento del número de árboles permite una mayor **diversidad** de los mismos, y por lo tanto **reduce el error OOB**. Sin embargo, la mejora se **estanca** a partir de un determinado número de árboles

- El mecanismo de construcción de random forest permite establecer un **baremo** de la **importancia** de cada **variable** en la **predicción final**.
- Para ello se calcula el error de la muestra **Out of the bag** y posteriormente se permutan elementos para calcular el error.
- Así las variables menos importantes deberían alterar menos la diferencia entre el error de la muestra OOB y el error de la muestra OOB permutada, que las variables importantes



- El algoritmo random forest puede ser utilizado para medir el aislamiento de los datos
- Dados dos elementos (i,j) , la proximidad entre ambos puede ser medida como la fracción de árboles en los cuales los elementos i y j están en el mismo nodo terminal.
- Si se hace esta operación para todos los pares de elementos, se formará una matriz de proximidad entre los elementos. E



VENTAJAS

- **No** se necesita **poda**
- Buena **precision**
- Evitar **over-fitting**
- **Sencilla** parametrización
- Se genera **importancia** de las **variables**.

INCONVENIENTES

- **Variables** categóricas con **muchos niveles** son **favorecidas** en el algoritmo.
- Mantenimiento de los **rangos** de datos en problemas de regression
- Reducida **interpretabilidad**.

GRACIAS
