

ANÁLISIS Y APRENDIZAJE AUTOMÁTICO



IBERIA EXPRESS

Yvonne Gala García y Jesús Prada Alonso



Yvonne Gala García
**Responsable Advanced
Analytics en Iberia Express**

Estudios:

- Doctorado en Aprendizaje Automático en la Universidad Autónoma de Madrid (UAM).
- Máster universitario en Bioinformática y bioestadística, UOC.
- Máster Universitario en Investigación e Innovación en TIC, especialidad inteligencia computacional, UAM.
- Máster Educación Secundaria Obligatoria, especialidad matemáticas, UAM.
- Licenciatura en Matemáticas, UAM.

Experiencia Laboral:

- Gerente Marketing Digital en Iberia Express.
- Responsable Advanced Analytics en Iberia Express.
- Freelance Data Scientist: Iberia Express, Piperlab.
- Profesora en EDEM y DevAcademy.
- Investigadora en el Grupo de Aprendizaje Automático de la UAM.





Jesús Prada Alonso
Responsable Area Machine Learning en Sigesa y Data Scientist en Iberia Express

Estudios:

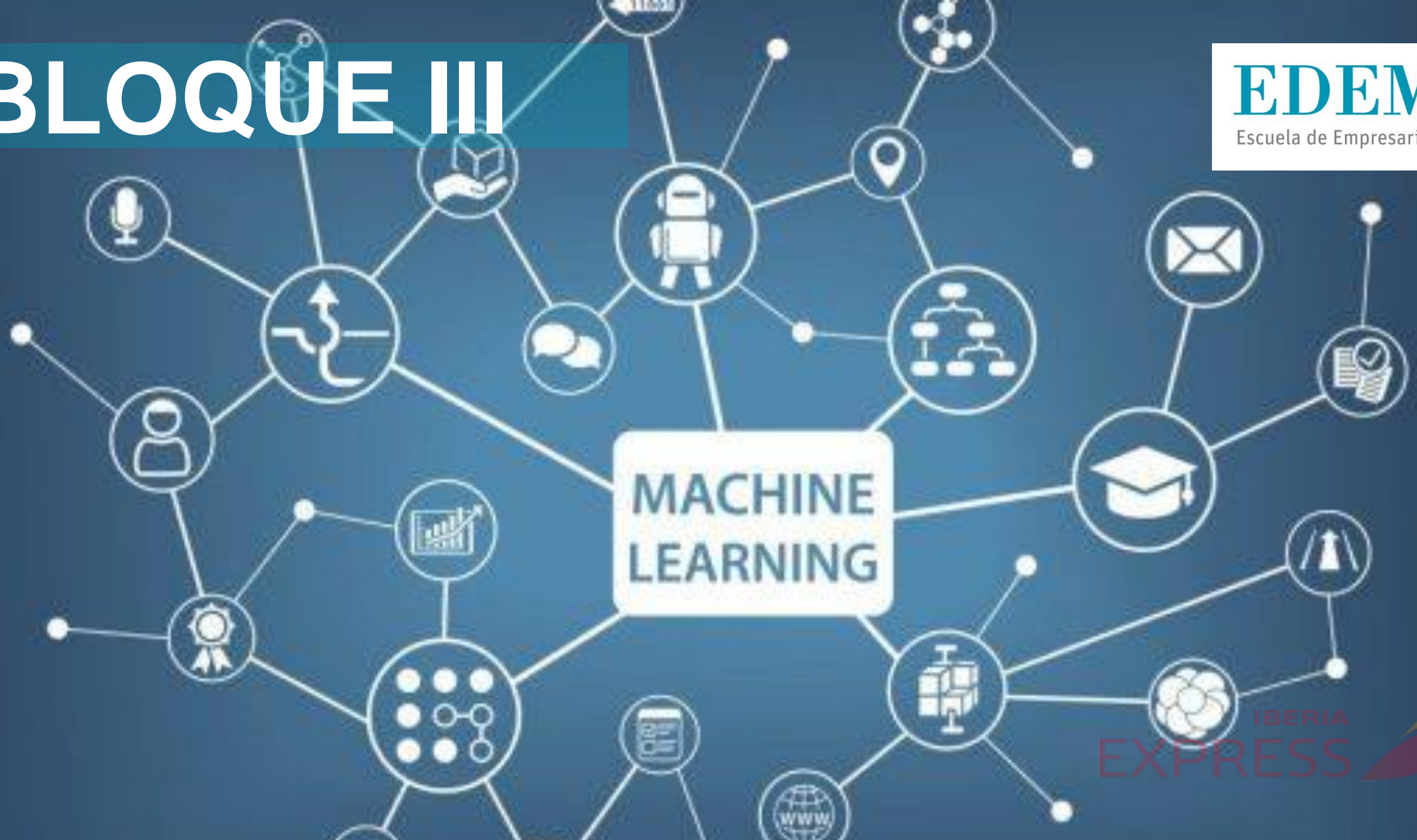
- Doctorado en Aprendizaje Automático en la Universidad Autónoma de Madrid (UAM).
- Máster universitario en Bioinformática y bioestadística, UOC.
- Doble Máster Universitario en Investigación e Innovación en TIC y en Matemáticas y Aplicaciones, UAM.
- Doble Titulación en Ingeniería Informática y Licenciatura en Matemática, UAM.

Experiencia Laboral:

- Responsable Area Machine Learning en Sigesa.
- Senior Data Scientist en Iberia Express
- Freelance Data Scientist: M+ Vision Consortium, Hospital Clínico San Carlos, UMOT, etc.
- Profesor en EDEM e IE School of Human Sciences and Technology, HST.
- Data Scientist en Kernel Analytics.
- Investigador en el Grupo de Aprendizaje Automático de la UAM.



BLOQUE III



BLOQUE III (09/04/2022): Modelos ML II, árboles de decisión, random forest y xgboost. Clasificación y Regresión.

TEORÍA

1. Árboles de decisión.
 - Conceptos.
 - Representación gráfica.
 - Hiperparámetros.
2. Random Forest.
 - Conceptos.
 - Ensembles de árboles.
 - Hiperparámetros.
3. XGBoost.
 - Conceptos.
 - Intuición Random Forest vs XGBoost.
 - Hiperparámetros.
 - Comparación. Interpretabilidad vs Precisión

BLOQUE III (09/04/2022): Modelos ML II, árboles de decisión, random forest y xgboost. Clasificación y Regresión.

PRÁCTICA:

Datasets:







- Dataset scikit-learn.
- Datos médicos.
- Datos aerolínea (Iberia Express).

Ejercicios:




- Script de árboles de decisión.
- Random forest para predicción de demanda
- Xgboost para predicción de exitus,.
- Problema de clasificación en medicina. Comparativa de modelos.
- Ejercicio: Problema de regresión en una aerolínea. Comparativa de modelos.



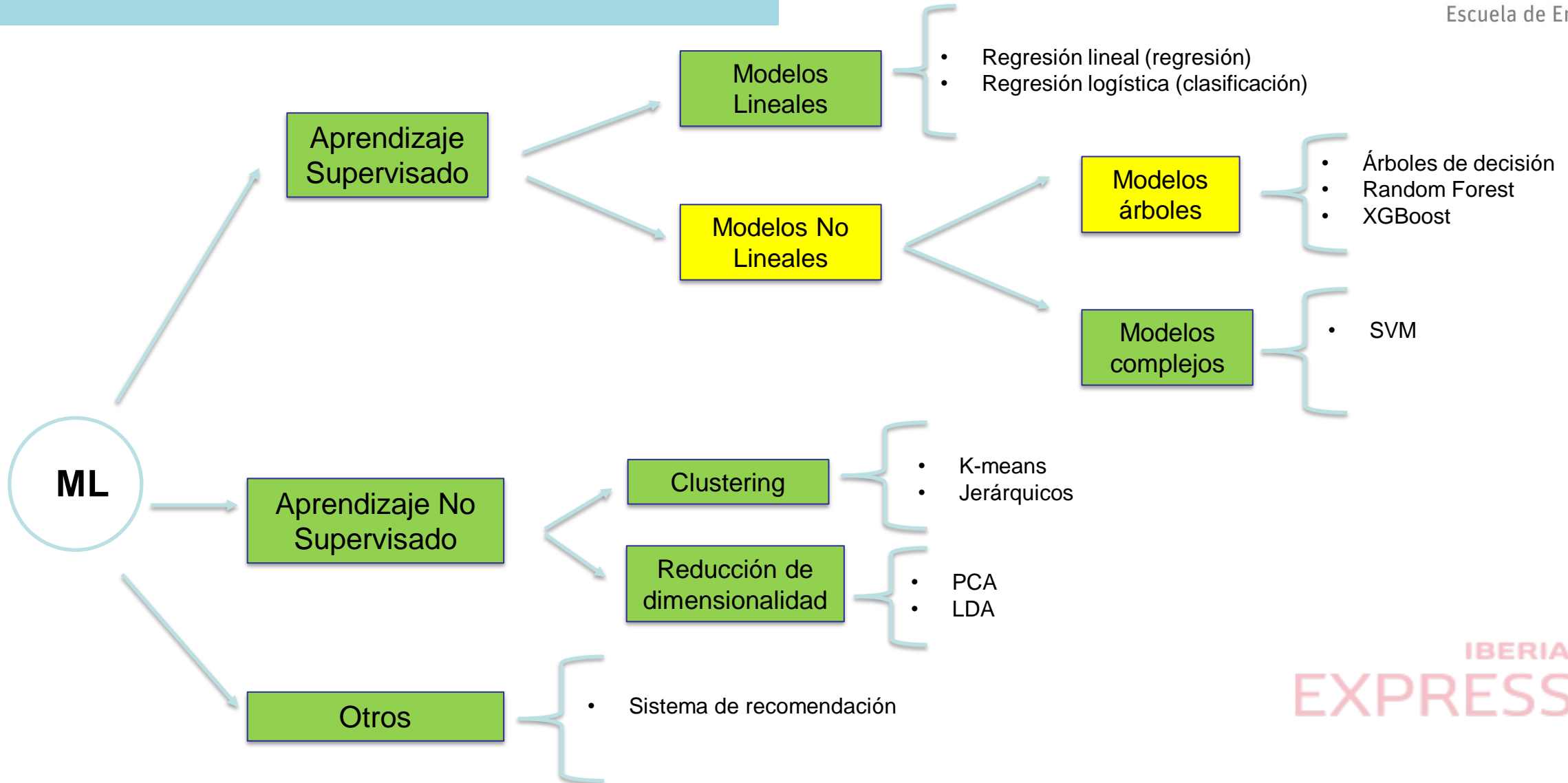
COMPARACIÓN ALGORITMOS

	TYPE	NAME	DESCRIPTION	ADVANTAGES	DISADVANTAGES
Linear		Linear regression	The "best fit" line through all data points. Predictions are numerical.	Easy to understand -- you clearly see what the biggest drivers of the model are.	<ul style="list-style-type: none"> ✗ Sometimes too simple to capture complex relationships between variables. ✗ Tendency for the model to "overfit".
		Logistic regression	The adaptation of linear regression to problems of classification (e.g., yes/no questions, groups, etc.)	Also easy to understand.	<ul style="list-style-type: none"> ✗ Sometimes too simple to capture complex relationships between variables. ✗ Tendency for the model to "overfit".
Tree-based		Decision tree	A graph that uses a branching method to match all possible outcomes of a decision.	Easy to understand and implement.	<ul style="list-style-type: none"> ✗ Not often used on its own for prediction because it's also often too simple and not powerful enough for complex data.
		Random Forest	Takes the average of many decision trees, each of which is made with a sample of the data. Each tree is weaker than a full decision tree, but by combining them we get better overall performance.	A sort of "wisdom of the crowd". Tends to result in very high quality models. Fast to train.	<ul style="list-style-type: none"> ✗ Can be slow to output predictions relative to other algorithms. ✗ Not easy to understand predictions.
		Gradient Boosting	Uses even weaker decision trees, that are increasingly focused on "hard" examples.	High-performing.	<ul style="list-style-type: none"> ✗ A small change in the feature set or training set can create radical changes in the model. ✗ Not easy to understand predictions.
Support Vector		SVM	Creates a hyperplane with maximum margin.	Can handle extremely complex tasks	<ul style="list-style-type: none"> ✗ Slow to train ✗ Difficult to understand

COMPARACIÓN ALGORITMOS

Tree-based		Decision tree	A graph that uses a branching method to match all possible outcomes of a decision.	Easy to understand and implement.	X Not often used on its own for prediction because it's also often too simple and not powerful enough for complex data.
		Random Forest	Takes the average of many decision trees, each of which is made with a sample of the data. Each tree is weaker than a full decision tree, but by combining them we get better overall performance.	A sort of "wisdom of the crowd". Tends to result in very high quality models. Fast to train.	X Can be slow to output predictions relative to other algorithms. X Not easy to understand predictions.
		Gradient Boosting	Uses even weaker decision trees, that are increasingly focused on "hard" examples.	High-performing.	X A small change in the feature set or training set can create radical changes in the model. X Not easy to understand predictions.

RESUMEN MODELOS



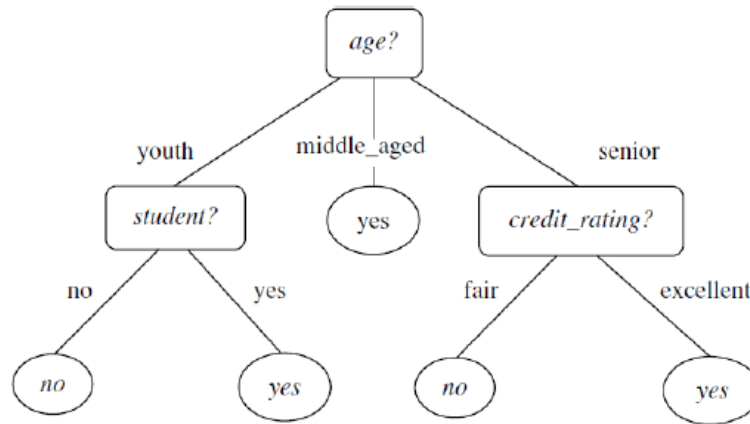
ÁRBOLES DE DECISIÓN



- Definición.
- Construcción del árbol.
 - Reglas de división.
 - Criterios de parada.
- Overfitting y poda.
- Hiperpámetros.
- Pros y contras.

Definición (I)

Este tipo de modelos representa el algoritmo en forma de árbol donde cada parte del atributo es un nodo que se bifurca en los posibles valores.

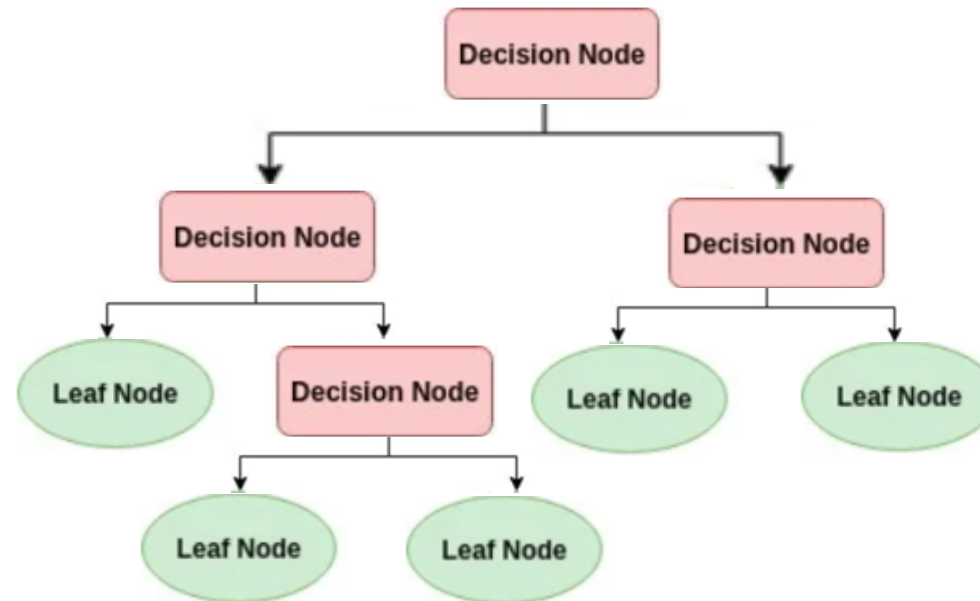


Consideraciones:

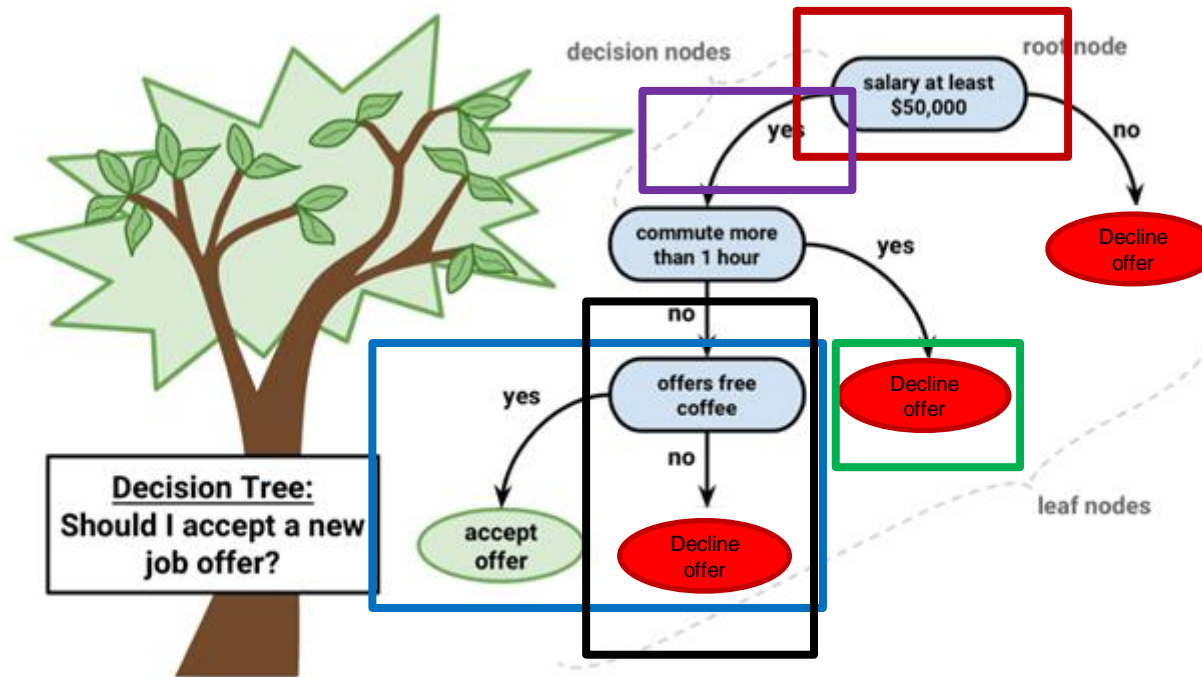
- Cada nodo empezando por el nodo raíz responde a una pregunta sobre un único atributo o variable.
- Cada rama corresponde a los valores que puede tomar dicho atributo.
- Los nodos hojas contienen los valores del target.

Definición (II)

- Cada nodo representa las variables de los datos, cada rama una decisión y cada nodo hoja un output.
- El nodo superior es el nodo raíz y representa la variable más importante.
- El árbol va aprendiendo de la división de los nodos recursivamente.
- Cuando un nuevo dato debe ser clasificado recorre el árbol desde el nodo raíz al nodo hoja, respondiendo a las preguntas de cada nodo en función de los atributos y el camino correcto en cada respuesta.



Definición (III)



1. Nodo raíz: población completa o muestra
2. Rama (decisión)
3. Nodos padre e hijo
4. Nodo terminal y hoja
5. Rama/sub-árbol

Clasificación VS Regresión

La diferencia entre árboles de decisión para clasificación y para regresión es la siguiente:

Regresión	Clasificación
Variable dependiente es continua	Variable dependiente es categórica
Valores de los nodos terminales se reducen a la media de las observaciones en esa región.	El valor en el nodo terminal se reduce a la moda de las observaciones del conjunto de entrenamiento que han “caído” en esa región.

Construcción (I)

1. Seleccionamos el mejor atributo usando el algoritmo Attribute Selection Measure (**Criterio de selección**).
2. **Separamos cada atributo** en segmentos.
3. Se repite el proceso hasta que se cumple algunas de las condiciones (**Criterio de parada**)

El algoritmo se inicializa con D (el conjunto de datos de entrenamiento), la lista de atributos y el método de selección de atributos S (la función que calcula el mayor beneficio).

1. Se crea el nodo N.
2. Si las tuplas en D tienen todas la misma clase C, return N como un nodo hoja etiquetado con la clase C.
3. En otro caso, si la lista de atributos está vacía, return N como un nodo hoja etiquetado con la clase mayoritaria en D.
4. En otro caso:
 1. Aplicar el método de selección de atributos sobre los datos y la lista de atributos, para encontrar el mejor atributo actual A: $S(D, \text{lista de atributos}) \rightarrow A$.
 2. Etiquetar a N como A y eliminar A de la lista de atributos.
 3. Para cada valor a_j del atributo A:
 1. Separar todas las tuplas en D para las cuales el atributo A toma el valor a_j , creando el subconjunto de datos D_j .
 2. Si D_j está vacío, añadir a N un nodo hoja etiquetado con la clase mayoritaria en D.
 3. En otro caso, añadir a N el nodo hijo resultante de llamar a **Generar_Arbol (Dj, lista de atributos)**.
 4. Return N.

Reglas de selección y división

Índice Gini (CART, SLIQ, SPRINT):

- La particularidad del índice Gini es que sólo crea árboles binarios, es decir, de cada nodo sólo pueden salir dos ramas correspondientes a los valores yes y no.
- **Mide el grado de impureza** de los nodos, cuán desordenados quedan los nodos una vez divididos.
- *Deberemos **minimizar** ese GINI index.*

Ganancia de información (ID3, C4.5):

- Se utiliza para **atributos categóricos** (cómo en hombre/mujer). Todos los atributos continuos de un conjunto de datos deben ser **discretizados**.
- **Estima la información que aporta cada atributo** basado en teoría de la información.
- Se basa en el concepto de **entropía**. Al obtener la medida de entropía de cada atributo, podemos calcular la ganancia de información del árbol.
- *Deberemos **maximizar** esa ganancia.*

Existen otras muchas reglas de división: χ^2 , MDL (Minimum Minimum Description Description Length)....

Reglas de división. Índice Gini (I)

Cuanto más impuro es un nodo más información requiere para ser descrito.

$$\text{Gini}(D) = 1 - \sum_{i=1}^m p_i^2$$

p_i es la probabilidad de que un ejemplo de D pertenezca a la clase C_i

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

En D hay 14 datos, de las cuales 9 se clasifican yes y 5 no.

- La probabilidad de que un dato sea yes es 9/14.
- La probabilidad de que un dato sea no es 5/14.

$$\text{Gini}(D) = 1 - \sum_{i=1}^m p_i^2 = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$

Reglas de división. Índice Gini (I)

El índice Gini considera un split binario de cada variable y suma la impureza de cada partición.

$$\Delta Gini(A) = Gini(D) - Gini_A(D) = 0.459 - Gini_A(D), \text{ donde se cumple: } Gini_A(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)$$

Para variables continuas o no binarias, se prueban distintos puntos de división y aquel con un índice de Gini más pequeño es el elegido.

R/D	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

El atributo edad tiene los valores, joven, adulto, senior.

- Cuando age=joven (5/14) → 2 son yes y 3 no.
- Cuando age=adulto (4/14) → 4 son yes y 0 no.
- Cuando age=senior (5/14) → 3 son yes y 2 no.

Senior o joven (10/14) → 5 son yes y 5 no.

Adultos (4/14) → 4 yes y 0 no.

$$Gini_{age\{youth, senior\}} = \frac{10}{14} \left(1 - \left(\frac{5}{10} \right)^2 - \left(\frac{5}{10} \right)^2 \right) + \frac{4}{14} \left(1 - \left(\frac{4}{4} \right)^2 - \left(\frac{0}{4} \right)^2 \right) = 0.357$$

$$Gini_{income\{low, medium\}} = 0.443, Gini_{income\{medium, high\}} = 0.45, Gini_{\{low, high\}} = 0.458,$$

$$Gini_{student} = 0.367, Gini_{credit_rating} = 0.429$$

La edad minimiza el índice Gini. → $Gini_A(D) = 0.357$

Juntando este resultado con el obtenido anteriormente tenemos:

$$\Delta Gini(A) = Gini(D) - Gini_A(D) = 0.459 - 0.357 = 0.102$$

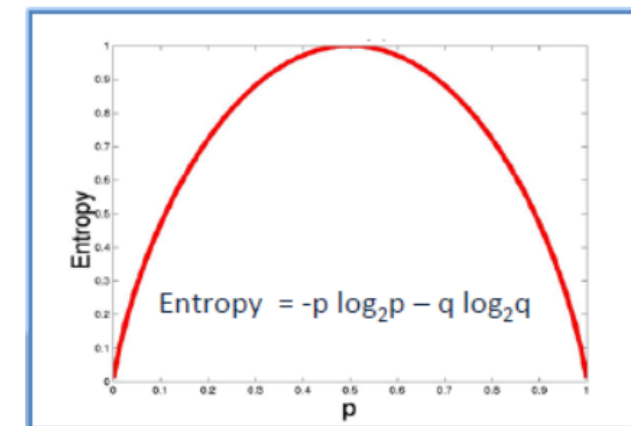
Reglas de división. Ganancia de información. ID3 (I)

La entropía se define como la información necesaria para clasificar un ejemplo D.

$$Info(D) = -\sum_{i=1}^m p_i \log_2(p_i)$$

P_i es la probabilidad de que un ejemplo de D pertenezca a la clase C_i

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no



$$Entropy = -0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$$

En D hay 14 datos, de las cuales 9 se clasifican yes y 5 no.

- La probabilidad de que un dato sea yes es 9/14.
- La probabilidad de que un dato sea no es 5/14.

$$Info(D) = -\sum_{i=1}^m p_i \log_2(p_i) = -\frac{9}{14} \log_2\left(\frac{9}{14}\right) - \frac{5}{14} \log_2\left(\frac{5}{14}\right) = 0.940 \text{ bits.}$$

Información necesaria para clasificar la tupla como $C_1 = \text{yes}$

Información necesaria para clasificar la tupla como $C_2 = \text{no}$

Info(D) es la información **media** necesaria para clasificar una tupla en C

Reglas de división. Ganancia de información. ID3 (I)

Información necesaria para clasificar D después de usar el atributo A para dividir D en v particiones.

$$Info_A(D) = \sum_{j=1}^v \frac{|D_j|}{|D|} \times I(D_j) \quad \text{donde } I(D) \text{ es la fórmula anterior} \quad Info(D) = - \sum_{i=1}^m p_i \log_2(p_i)$$

|D| es el número de patrones y |D_j| es el número de patrones donde el atributo A toma el valor A_j, $\frac{|D_j|}{|D|}$ es el peso de la partición j

Finalmente la ganancia de información se define como: $Gain(A) = Info(D) - Info_A(D)$

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

El atributo edad tiene los valores, joven, adulto, senior.

- Cuando age=joven (5/14) → 2 son yes y 3 no.
- Cuando age=adulto (4/14) → 4 son yes y 0 no.
- Cuando age=senior (5/14) → 3 son yes y 2 no.

$$Info_{age}(D) = \frac{5}{14} \left(-\frac{2}{5} \log_2 \left(\frac{2}{5} \right) - \frac{3}{5} \log_2 \left(\frac{3}{5} \right) \right) + \frac{4}{14} \left(-\frac{4}{4} \log_2 \left(\frac{4}{4} \right) \right) + \frac{5}{14} \left(-\frac{3}{5} \log_2 \left(\frac{3}{5} \right) - \frac{2}{5} \log_2 \left(\frac{2}{5} \right) \right) = 0.694 \text{ bits}$$

$$Gain(age) = Info(D) - Info_{age}(D) = 0.940 - 0.694 = 0.246 \text{ bits}$$

$$Gain(income) = Info(D) - Info_{income}(D) = 0.029 \text{ bits}$$

$$Gain(student) = Info(D) - Info_{student}(D) = 0.151 \text{ bits}$$

$$Gain(credit_rating) = Info(D) - Info_{credit_rating}(D) = 0.048 \text{ bits}$$

Reglas de división. Comparación

Ganancia de Información:

- Sesgado a atributos con muchos valores diferentes.

Índice de Gini:

- Funciona peor si tiene muchas clases.
- Favorece particiones de tamaño y pureza similares.

No existe ninguna técnica significativamente mejor.

Criterio de parada

Se repite el proceso de selección y división hasta alcanzar algunos de los criterios de parada.

- Cuando todos los ejemplos que quedan **pertenecen a la misma clase** (se añade una hoja al árbol con la etiqueta de la clase).
- Cuando **no quedan atributos por los que ramificar** (se añade una hoja etiquetada con la clase más frecuente en el nodo).
- Cuando **no nos quedan datos que clasificar** (no hay más patrones).

Overfitting – Poda (I)

- Uno de los **problemas de los árboles** de decisión es que tienden a ajustarse demasiado a los datos (**overfitting**).
- Demasiadas ramas es un **indicador de** anomalías en los datos, **outliers** o ruido.
- Para evitarlo existen **técnicas de poda** que consisten en eliminar ramas o no dejar que el árbol crezca.
- Existen varias aproximaciones:
 - **Prepoda**: durante el crecimiento del árbol.
 - **Postpoda**: después de la creación del árbol.

Overfitting – Poda (II)

Prepoda:

- Se aplica el criterio según va creciendo el árbol.
- El objetivo es detener el crecimiento del árbol.
 - Si un nodo solo tiene ejemplos de una clase.
 - Establecer una cota de profundidad.
 - Se construye un nodo con menos de n patrones.

Parámetros de la Poda o Crecimiento del árbol

- **Mínimo de observaciones para dividir un nodo**
Mínimo número de muestras que se requieren en un nodo para ser considerado para ramificación.
- **Mínimo número de observaciones para un nodo terminal**
Valores más bajos son necesarios para problemas de clases no balanceadas.
- **Máxima profundidad del árbol (vertical)**
Una mayor profundidad permite aprender relaciones más específicas.
- **Máximo número de nodos hoja**
Se puede definir en lugar de máxima profundidad. Profundidad n = máximo 2^n hojas.
- **Máximo número de atributos a considerar para la ramificación**
Por defecto, se utiliza la raíz cuadrada del número total de atributos.

Overfitting – Poda (II)

Postpoda: Una vez construido se sustituyen subárboles.

- Un subárbol por un nodo hoja, que será la clase más frecuente del subárbol.
- Un subárbol por otro más pequeño.

Parámetro de complejidad

El proceso de poda usa una medida que combina el error o coste de la predicción y la complejidad.

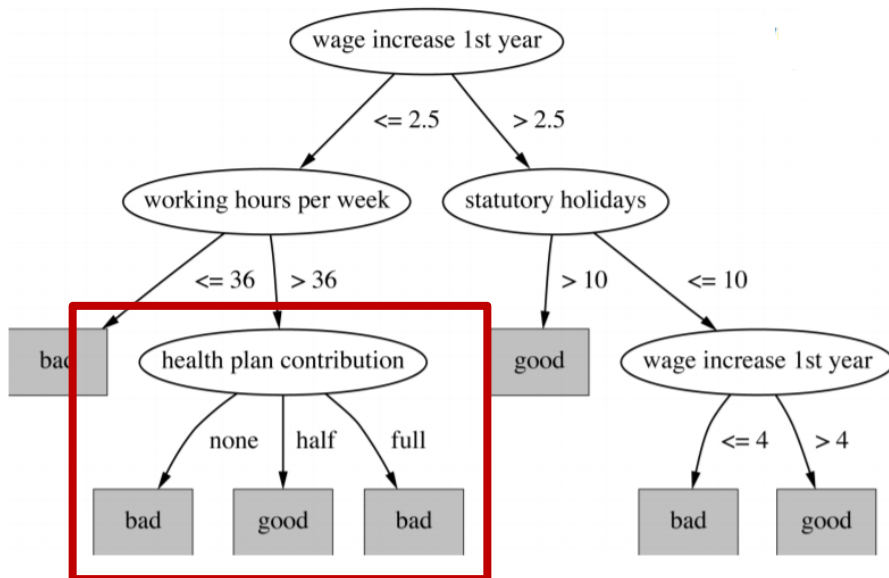
$$R_{\alpha}(T) = R(T) + \alpha |\tilde{T}|$$

Donde $R(T)$ nos da el error de clasificación o la suma de las varianzas residuales en regresión.

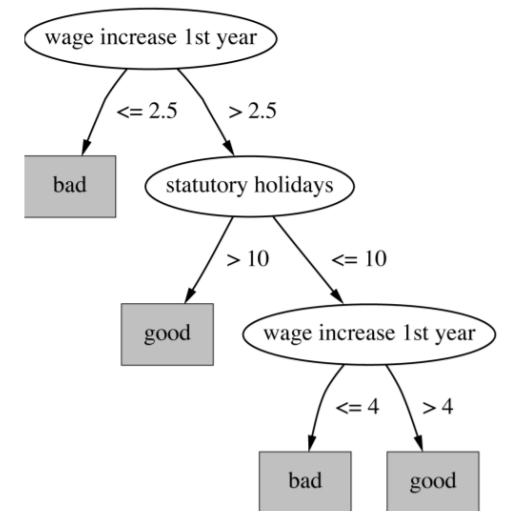
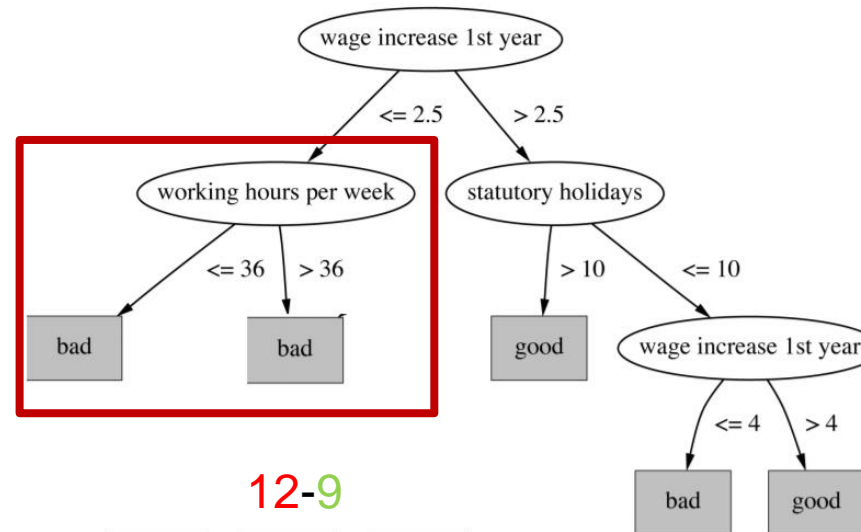
$\alpha |\tilde{T}|$ penaliza los árboles complejos, siendo α el parámetro de complejidad

El parámetro α es análogo al λ en Lasso/Ridge

Overfitting – Poda (III)



3-2 5-1 8-2



Parámetros del árbol. Scikitlearn.

Nombre	Utilidad	Valor por defecto
criterion	criterio utilizado como regla de división	Gini
min_samples_split	cantidad mínima de muestras que debe tener un nodo para poder subdividir	2
min_samples_leaf	cantidad mínima que puede tener una hoja final. Si tuviera menos, no se formaría esa hoja y “subiría” al nodo padre	1
max_depth	Es la profundidad máxima del árbol, controlar crecimiento y complejidad. Si el parámetro es None el árbol se expandirá hasta que todas las hojas son puras o se cumple el min_samples_Split	None
max_features	número de variables para elegir el mejor split	None

Ventajas y Desventajas

Pros

- Son **fáciles de interpretar** y representar.
- Puede interpretar **patrones no lineales**.
- Puede usarse con **datos categóricos (aunque depende de la implementación)**.
- **No requiere mucho tratamiento de datos**, por ejemplo no es tan necesario normalizar.
- Nos da una **medida de importancia de variables**.
- Los árboles de decisión no asumen ninguna distribución en los datos.

Ventajas y Desventajas (II)

Contras

- Es muy sensible a datos con ruido y puede dar **overfitting**.
- Una **pequeña variación en los datos puede dar un árbol de decision diferente**. Esto puede mejorar mediante técnicas de bagging o boosting que combinan varios árboles.
- Son modelos con **mucho bias** si los **datos están desbalanceados**.
- Los **resultados son muy inferiores a las de modelos más complejos** como SVM o redes neuronales.

RANDOM FOREST

EDEM

Escuela de Empresarios



IBERIA
EXPRESS

- Introducción.
- Ensembles.
- Random Forest VS árboles de decisión.
- Construcción de random forest.
 - Muestreo.
 - Votación.
- Hiperpámetros.
- Importancia de variables.
- Pros y contras.

Definición

- Random Forest es un algoritmo de aprendizaje supervisado que puede ser utilizado tanto para **clasificación** como para **regresión**.
- Es un **ensemble** de **árboles de decisión**.
- Cuantos más árboles tiene más robusto es el algoritmo.
- Se crean árboles con un subconjunto aleatorio de los datos.
- Se hace la predicción de cada árbol y se selecciona el que más votos tiene (clasificación) o la media (regresión).
- Nos proporciona de forma directa un buen indicador de importancia de las variables.

Métodos de ensemble

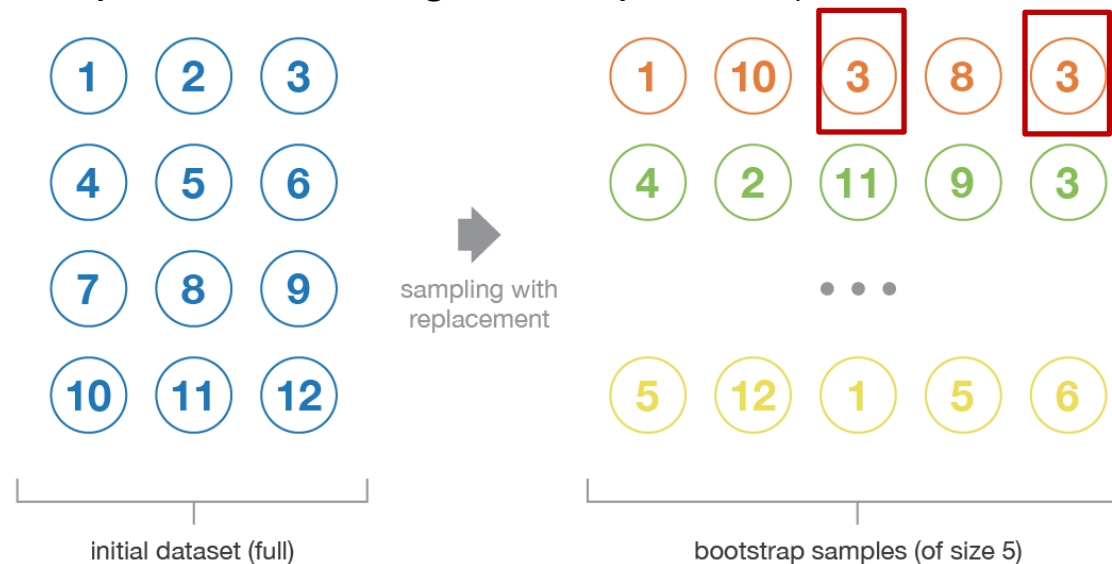
- Se basan en construir un modelo que es a su vez una combinación de modelos.
- Estos modelos individuales suelen llamarse **weak learners** y el modelo final **strong learner**.
- Los modelos individuales pueden sufrir de high bias o high variance, pero el modelo final no.
- Tres métodos principales:
 - Bagging.
 - Boosting.
 - Stacking.

Bagging (bootstrap aggregating). Características

- **Weak learners homogéneos.** Es decir, combina modelos de una misma familia. Ej: Árboles de decisión.
- Estos modelos individuales se entrenan **en paralelo**.
- Trata de combinar **modelos individuales complejos** (high variance). Ej: Árboles profundos.
- Combina las predicciones de los modelos individuales usando algún tipo de valor promedio/votación, reduciendo su varianza. Cada hipótesis tiene el mismo peso en la votación de la predicción final.
- Para optimizar cada weak learner se usa un muestreo aleatorio del conjunto original (**bootstrap**).
- Ejemplo: **Random forest**.

Bagging. Bootstrapping

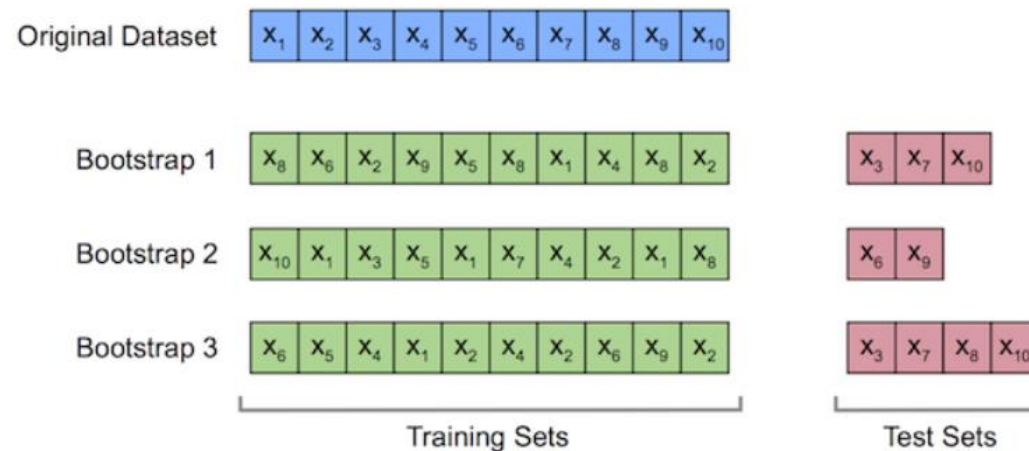
- Se basa en el uso del **bootstrapping**.
- La idea es generar subconjuntos aleatorios de tamaño B a partir del dataset inicial de tamaño N.
- En estos subconjuntos puede haber registros repetidos (**muestreo con repetición**).



- La N debe ser suficientemente grande (representatividad) y mayor que B (independencia).

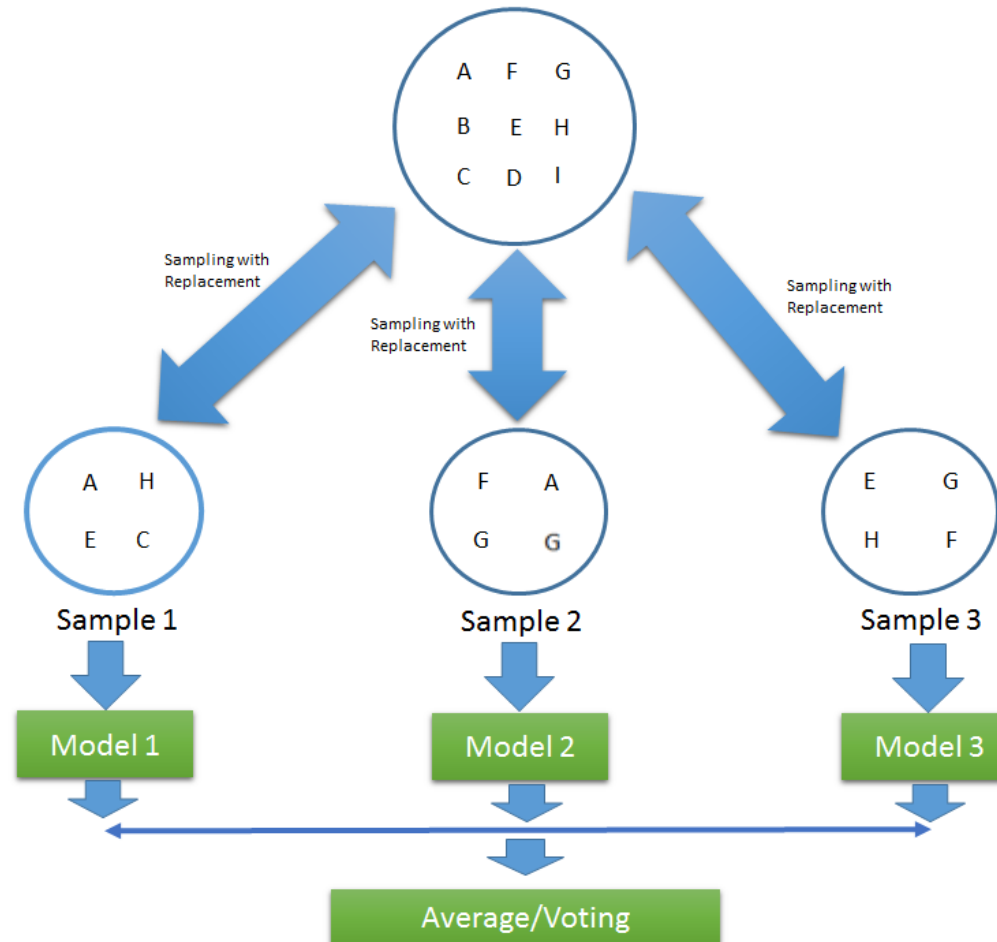
Bagging. Bootstrapping (II)

- Un porcentaje de los datos del dataset inicial no se usan para el entrenamiento y pueden ser usados para validación.
- Este conjunto se denomina **out of bag (OOB)** samples.



- El error estimado en estos out of bag samples se conoce como OOB error.

Bagging. Esquema



Boosting. Características

- **Weak learners homogéneos.** Ej: Árboles de decisión.
- Estos modelos individuales se entrenan de forma **secuencial**.
- Trata de combinar **modelos individuales sencillos** (high bias). Ejemplo: Árboles poco profundos.
- Cada nuevo modelo individual busca corregir los errores de los anteriores, reduciendo su bias.
- La secuencialidad se compensa con el hecho de entrenar modelos individuales menos costosos.
- Tienen más riesgo de overfitting que los modelos de bagging, pero un mayor potencial predictivo.
- Ejemplos: AdaBoost, **XGBoost**.

Boosting. AdaBoost

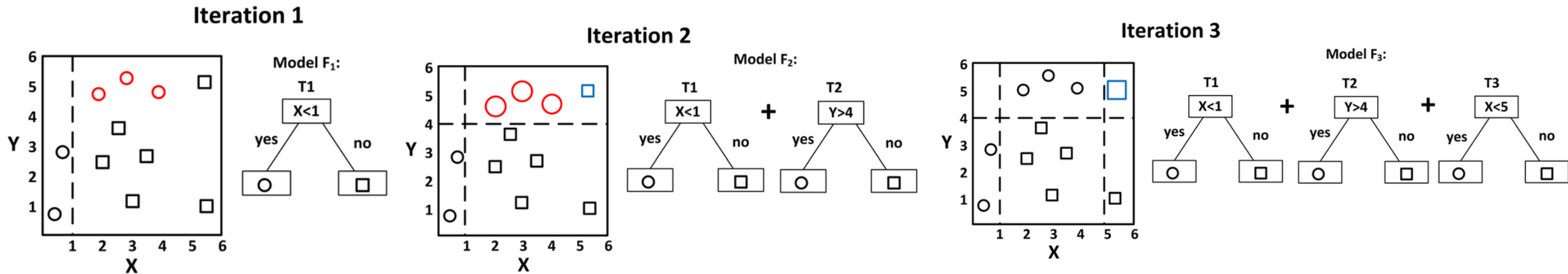
- El modelo final (strong learner, S) se define como:

$$S = \sum_{i=1}^L c_i I_i$$

donde I_i es cada uno de los modelos individuales y c_i sus coeficientes.

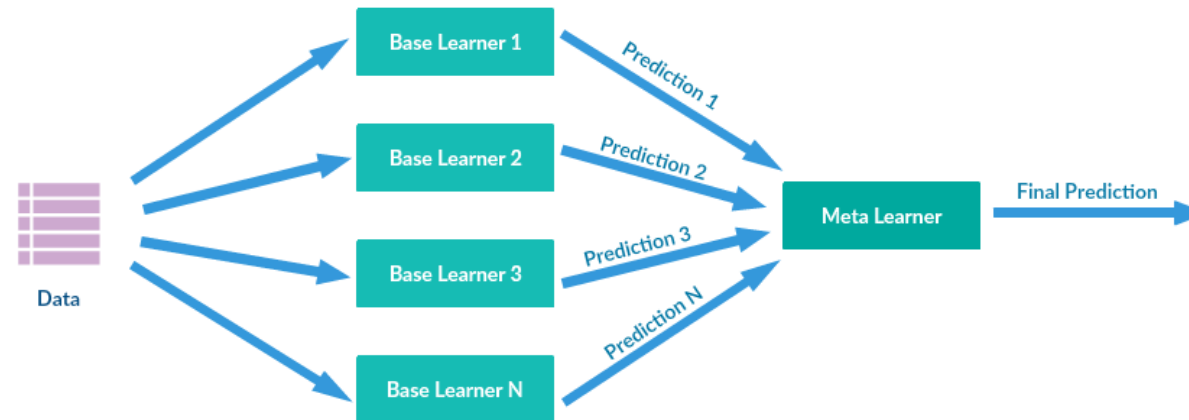
- **Algoritmo:**
 1. Se asigna a cada registro un peso inicial w . Inicialmente este peso es constante ($\frac{1}{N}$).
 2. Se entrena un modelo individual teniendo en cuenta los pesos w .
 3. Se añade el nuevo modelo individual a nuestro modelo final, con $c_i = \frac{1}{ERROR}$.
 4. Se actualizan los pesos aumentando w para aquellos registros donde el modelo agregado S falla más.
 5. Se repiten los pasos 2-4 hasta que el modelo converja o se llegue al número máximo de iteraciones.

Boosting. Esquema



Stacking. Características

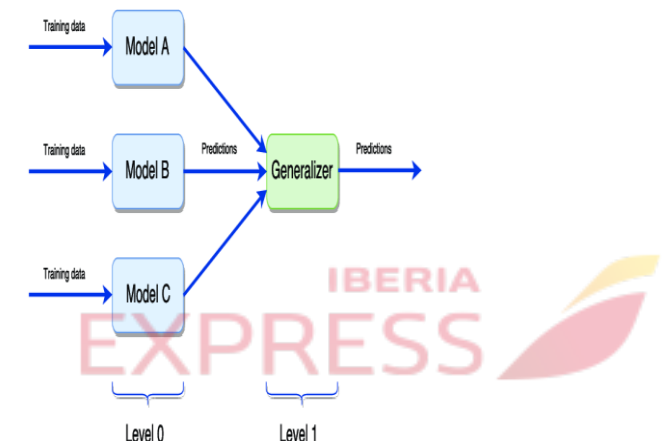
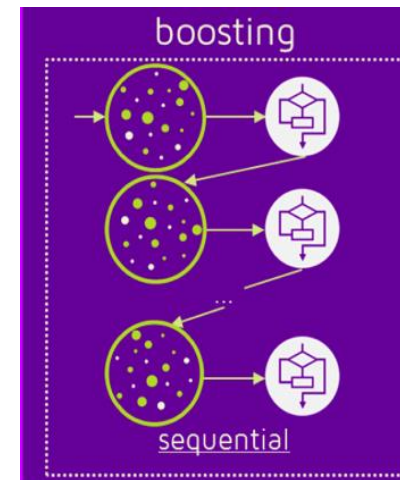
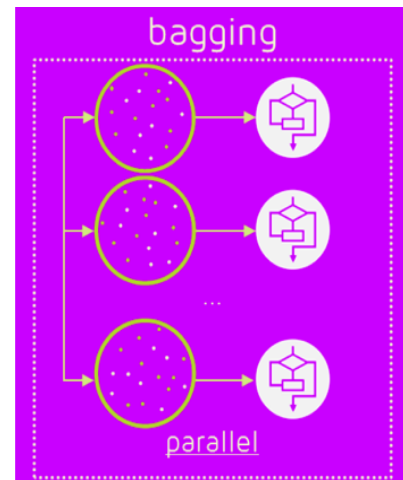
- Utiliza weak learners **heterogéneos**. Es decir, combina familias de modelos distintas.
- Estos modelos individuales se entrenan **en paralelo**, no se necesita un proceso secuencial.
- Las predicciones de cada modelo individual se introducen como variables predictoras o input del modelo final, también llamado **meta learner**.



Métodos de ensemble. Resumen

Característica	Bagging	Boosting	Stacking
Naturaleza Weak Learners	Homogéneos	Homogéneos	Heterogéneos
Flujo Entrenamiento	En paralelo	Secuencial	En paralelo
Tipo Weak Learners	Complejos (high variance)	Sencillos (high bias)	Variado
Combinación	Promedio	Pesos/coeficientes	Meta-modelo

Random Forest



Random Forest vs arboles de decisión

ÁRBOLES DE DECISIÓN

Es un modelo que es una representación gráfica de un árbol

Tienen riesgo de overfitting

Los resultados son menos precisos

Son modelos sencillos, fáciles de interpretar y visualizar

RANDOM FOREST

Es un modelo que con métodos de ensemble (bagging) construye múltiples árboles de decisión

Menos riesgo de overfitting

Los resultados son más precisos

Son modelos más complejos y más difíciles de interpretar y visualizar

Construcción

1. MUESTREO (Bootstrap):

- Supongamos que el número de casos en el conjunto de entrenamiento es N .
- Para cada árbol, una **muestra** de esos N casos se toma **aleatoriamente** pero **CON REEMPLAZO**.

2. SELECCIÓN DE VARIABLES:

- Si existen M variables de entrada, un número $m < M$ se especifica tal que **para cada nodo**, m variables se seleccionan aleatoriamente de M .
- El valor m se mantiene constante durante la generación de todo el bosque.

3. CRECIMIENTO:

- Cada árbol crece hasta su máxima extensión posible y **NO** hay proceso de **postpoda**.

4. VOTACIÓN:

- Nuevas instancias se predicen a partir de la **agregación de las predicciones de los x árboles** (mayoría/ratio de votos para clasificación, promedio para regresión)

Importancia de variables

- Al entrenar un modelo de random forest se calcula automáticamente una **importancia** o score **para cada variable**. Este valor se escala para que la suma de todas las variables sea 1.
- Este score nos puede ayudar a **seleccionar** las **variables** con mayor importancia.
- Esta selección de variables suele ser una **buena elección para modelos de la misma familia** (random forest). Sin embargo, para otras familias (ej: SVM) puede no serlo.
- Scikit-learn tiene un parámetro opcional para mostrar esta importancia de variables.
- Usa el algoritmo gini importance or **mean decrease in impurity** (MDI).
- Nos da información de cuánto de preciso va a ser el modelo si quitas una variable. Cuánto más decrezca la precision más importante es la variable.

Importancia de variables. Algoritmo

1. Para cada árbol en el bosque, calcular el número de votos correctos en la muestra Out Of Bag, *OOB_ERROR_ORIGINAL*.
2. Efectuar una permutación aleatoria de los valores de una variable en la muestra Out Of Bag, *OOB_ERROR_CAMBIO*.
3. Calcular el nuevo número de votos correctos con la variable modificada.
4. Calcular la diferencia entre los votos correctos antes y después de la permutación.

$$SCORE = OOB_ERROR_CAMBIO - OOB_ERROR_ORIGINAL$$

5. El promedio de este número sobre todos los árboles en el bosque es el “score” de importancia.

$$IMPORTANCIA = \sum SCORE$$

Hiperparámetros

- **n_estimators**: número de árboles en el bosque. Usar demasiados árboles puede ser innecesariamente ineficiente.
- **mtry**: número de variables aleatorias como candidatas en cada ramificación.
- **sampsize**: el número de muestras sobre las cuales entrenar. **Valores más bajos** podrían introducir **sesgo y reducir el tiempo**. **Valores más altos** podrían incrementar el rendimiento del modelo pero a **riesgo de causar overfitting**.
- **nodesize**: mínimo número de muestras dentro de los nodos terminales → También afecta al equilibrio entre bias-varianza.
- **maxnodes**: máximo número de nodos terminales.

El hiperparámetro más importante para ajustar es el número de árboles en el bosque (**n_estimators**) y el de variables candidatas a seleccionar para evaluar cada ramificación (**mtry**).

Hiperparámetros

Hiperparámetro	Efecto	Rango	Valor por defecto
Hiperparámetros árbol de decisión	Los mismos que en un árbol de decisión individual.	Ver sección anterior	Ver sección anterior
n_estimators	Número de árboles en el bosque. Usar demasiados árboles puede ser innecesariamente ineficiente.	$[1, \infty]$	100
max_features	Número de variables aleatorias como candidatas en cada ramificación.	$[1, \infty]$	$\sqrt{n_features}$
max_samples	El número de muestras sobre las cuales entrenar. Valores más bajos podrían introducir sesgo pero reducen el tiempo de computación. Valores más altos podrían incrementar el rendimiento del modelo pero a riesgo de causar overfitting.	$[None, \infty]$	None (dataset entero)
min_samples_leaf	Mínimo número de muestras dentro de los nodos terminales. También afecta al equilibrio entre bias-varianza.	$[1, \infty]$	1
max_leaf_nodes	Máximo número de nodos terminales.	$[None, \infty]$	None

Ventajas y Desventajas

- **Pros**

- Existen muy **pocas suposiciones a priori** sobre el conjunto de datos.
- Una de las salidas del modelo es la **importancia de variables**.
- Puede identificar las variables más significativas. Método de reducción de dimensionalidad.
- Incorpora métodos efectivos para **estimar valores no informados de forma automática**.
- Es posible usarlo como método no supervisado (**clustering**) y **detección de outliers**.
- **Robusto** frente a la selección de hiperparámetros.

*

- **Contras**

- **Pérdida de interpretación.**
- Bueno para clasificación, no tanto para regresión. **Las predicciones no son de naturaleza continua.**
- En regresión, no puede predecir más allá del rango de valores del conjunto de entrenamiento.

* También en un árbol de decisión, pero conclusiones estadísticamente menos significativas.

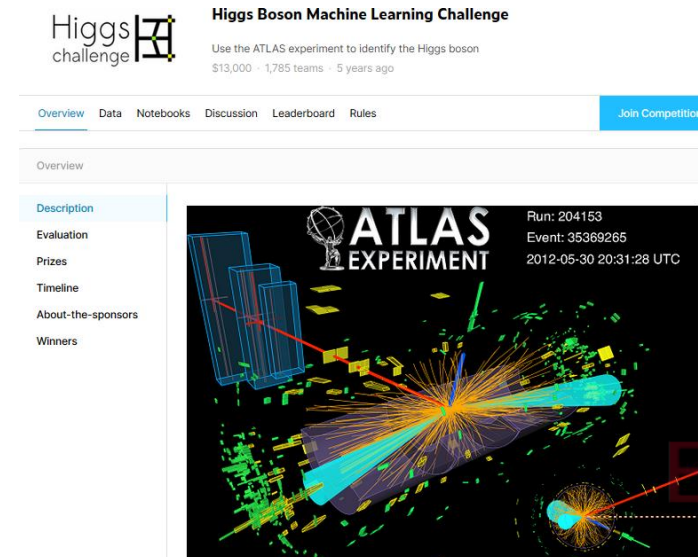
XGBOOST



- Introducción.
- Definición.
- Hiperpárametros de XGBoost.
- Interpretabilidad vs. Precisión.

INTRODUCCIÓN (I)

- Su nombre completo es **Extreme Gradient Boosting**.
- Se aplica a problemas de aprendizaje supervisado **tanto de clasificación como de regresión**.
- Es **una de las familias** de modelos ML **más potentes** en la actualidad, junto a SVMs y Deep Learning.
- **Empezó a hacerse conocida** tras su victoria en una competición de **Kaggle** sobre el Bosón de Higgs.
<https://www.kaggle.com/c/higgs-boson>.



XGBoost en Kaggle



- En las competiciones de 2019, 17 de las 29 (59%) soluciones ganadoras empleaban XGBoost.
- El segundo modelo más usado, redes neuronales profundas (Deep Learning) solo aparecía en 11 soluciones ganadoras.
- 8 de ellas empleaban solo XGBoost, 9 en combinación con otros modelos (stacking).
- Esto se debe a dos razones principales:
 - **Modelo con alto potencial predictivo.**
 - **Muy optimizado en tiempos y uso de memoria → escalable y permite probar muchos hiperparámetros.**

Gradient Descent

- En gradient descent cada iteración consiste en:

$$S_k = S_{k-1} - c_k \nabla_{k-1} E(S_{k-1})$$

donde:

- c_k es llamado step size (learning rate).
- $E(S_{k-1})$ son los errores del modelo en la iteración $k-1$.
- ∇ es el gradiente o derivada.
- Por tanto, $\nabla_{k-1} E(S_{k-1})$ es el gradiente del error del modelo en la iteración anterior.

De Gradient Descent a Gradient Boosting

$$S_k = S_{k-1} - c_k \nabla_{k-1} E(S_{k-1})$$

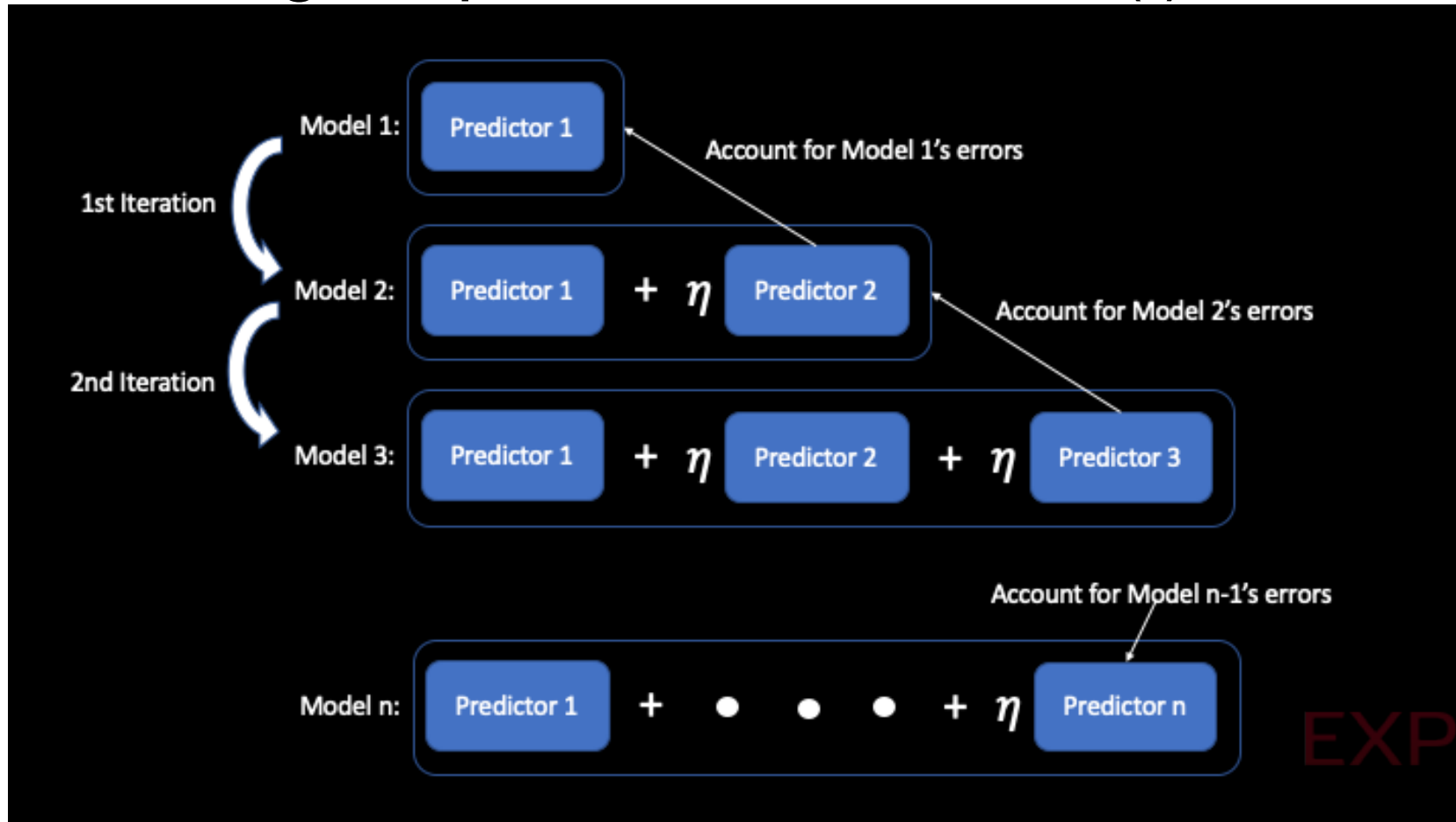
- En Gradient Boosting este proceso se adapta/simplifica a un enfoque de ensembles.
- En lugar de calcular el gradiente ∇ de los errores del modelo actual, $E(S_{k-1})$, se construye un nuevo weak learner que aprenda a predecir estos errores.
- Es decir: $\nabla_{k-1} E(S_{k-1}) \rightarrow I_k$, donde I_k es un modelo que busca predecir los errores del modelo S_{k-1} . Por tanto:

$$S_k = S_{k-1} - c_k I_k$$

donde:

- El símbolo $-$ indica que estamos corrigiendo los errores de S_{k-1} .
- c_k es el coeficiente del nuevo weak learner e indica cuánto queremos corregir los errores de S_{k-1} .

Gradient Boosting. Representación visual (I)



Algoritmo

- Resumiendo, Gradient Boosting sigue el siguiente algoritmo iterativo:
 1. Se inicializan los errores, E , con el valor del target a predecir. Por tanto, el primer weak learner predecirá el target.
 2. Se entrena un modelo individual que predice los errores E .
 3. Se añade el nuevo modelo individual a nuestro modelo final, con c_i el coeficiente que minimiza el error global del nuevo modelo agregado S_k .
 4. Se actualiza el valor de los errores $E = E(S_k)$.
 5. Se repiten los pasos 2-4 hasta que el modelo converja o se llegue al número máximo de iteraciones.

Ventajas de XGBoost

- XGBoost es una versión optimizada de los modelos clásicos de Gradient Boosting.
- Las principales razones de la popularidad de XGBoost frente a otros métodos de Boosting como AdaBoost o estándar Gradient Boosting es:
 - **Enfoque más complejo** que Adaboost basado en Gradient Descent.
 - Mayor número de hiperparámetros que permiten un **mayor fine-tuning** del modelo.
 - Mayor control de la regularización, lo que permite **evitar overfitting**.
 - **Muy optimizado** tanto en términos de coste computacional como de memoria → Altamente escalable.

Mayor potencial predictivo.

nthread

- Número de hilos a usar para paralelizar computaciones.
- Cuanto mayor sea más uso de CPU haremos pero más rápido entrenarán los modelos.
- **No es un hiperparámetro como tal** (no se debe incluir en la rejilla) pero conviene fijarlo en toda llamada a XGBoost para evitar saturar sistemas.
- Valor por defecto: Máximo número de hilos disponible.

nrounds

- Número de iteraciones de boosting a realizar.
- Los valores van de 1 a infinito.
- Valores altos llevan a modelos más complejos, por lo que pueden provocar overfitting.
- Valor por defecto: 100.

eta

- Inversamente relacionado con el step size, c_k .
- Los valores van de 0 a 1, con valores altos provocando actualizaciones más pequeñas y llevando a modelos más conservadores.
- Es el análogo del inverso del learning rate en un gradient descent.
- Valor por defecto: 0,3.

gamma

- Mínima mejora en la loss function requerida para realizar una partición adicional en un nodo hoja del árbol.
- Toma valores entre 0 e infinito.
- Cuanto mayor sea la gamma, más conservador será el algoritmo.
- Valor por defecto: 0 (no se exige mínima mejora).

max_depth

- Profundidad máxima de cada árbol.
- Toma valores entre 1 e infinito.
- Aumentar este valor hará que el modelo sea más complejo y tenga más probabilidades de overfitting.
- Valores altos también llevarán a modelos más lentos y, sobre todo, a un mayor coste en memoria.
- Valor por defecto: 6.

min_child_weight

- Suma mínima de pesos necesaria en un nodo hijo o terminal.
- Si el paso de partición del árbol da como resultado un nodo hoja con una suma de pesos menor que min_child_weight, entonces el proceso de construcción abandonará la partición adicional.
- Toma valores entre 0 e infinito.
- Cuanto mayor sea min_child_weight, más conservador será el algoritmo.
- Valor por defecto: 1.

subsample

- Ratio de submuestreo de las instancias de entrenamiento.
- Por ejemplo, establecerlo en 0.5 significa que XGBoost muestreará al azar la mitad de los datos de entrenamiento antes de entrenar cada modelo individual.
- El submuestreo ocurrirá una vez en cada iteración, de forma que modelos individuales distintos son entrenados con diferentes muestreos (bootstrapping sin repetición).
- Toma valores entre 0 y 1.
- Valores bajos disminuyen la probabilidad de overfitting, pero pueden provocar modelos con menor potencial predictivo al ver menos datos.
- Valor por defecto: 1 (no se hace submuestreo).

colsample_bytree

- Ratio de submuestreo de las columnas del dataset.
- Por ejemplo, establecerlo en 0.5 significa que XGBoost elegirá al azar la mitad de las columnas de los datos de entrenamiento antes de entrenar cada modelo individual.
- El submuestreo ocurrirá una vez en cada iteración, de forma que modelos individuales distintos son entrenados con diferentes selecciones de variables.
- Toma valores entre 0 y 1.
- Valores bajos disminuyen la probabilidad de overfitting, pero pueden provocar modelos con menor potencial predictivo al ver menos variables.
- Valor por defecto: 1 (no se hace submuestreo).

num_parallel_tree

- Número de árboles calculados en paralelo en cada iteración.
- Toma valores entre 1 e infinito.
- Valores mayores que uno implican que en lugar de tree boosting clásico se lleva a cabo random forest boosting.
- Valor por defecto: 1 (en cada iteración se construye un único árbol de decisión, no un random forest).

lambda

- Controla la regularización L2, o Ridge Regression, aplicada a los pesos.

$$l(\hat{y}_i, y_i) + \lambda \sum_{i=1}^d w_i^2$$

- Toma valores entre 0 e infinito.
- Valores mayores hacen el modelo más conservador y menos propenso a overfitting.
- Valor por defecto: 1.

alpha

- Controla la regularización L1, o Lasso, aplicada a los pesos.

$$l(\hat{y}_i, y_i) + \lambda \sum_{i=1}^d |w_i|$$

- Toma valores entre 0 e infinito.
- Valores mayores hacen el modelo más conservador y menos propenso a overfitting.
- Valor por defecto: 0 (no hay regularización L1).

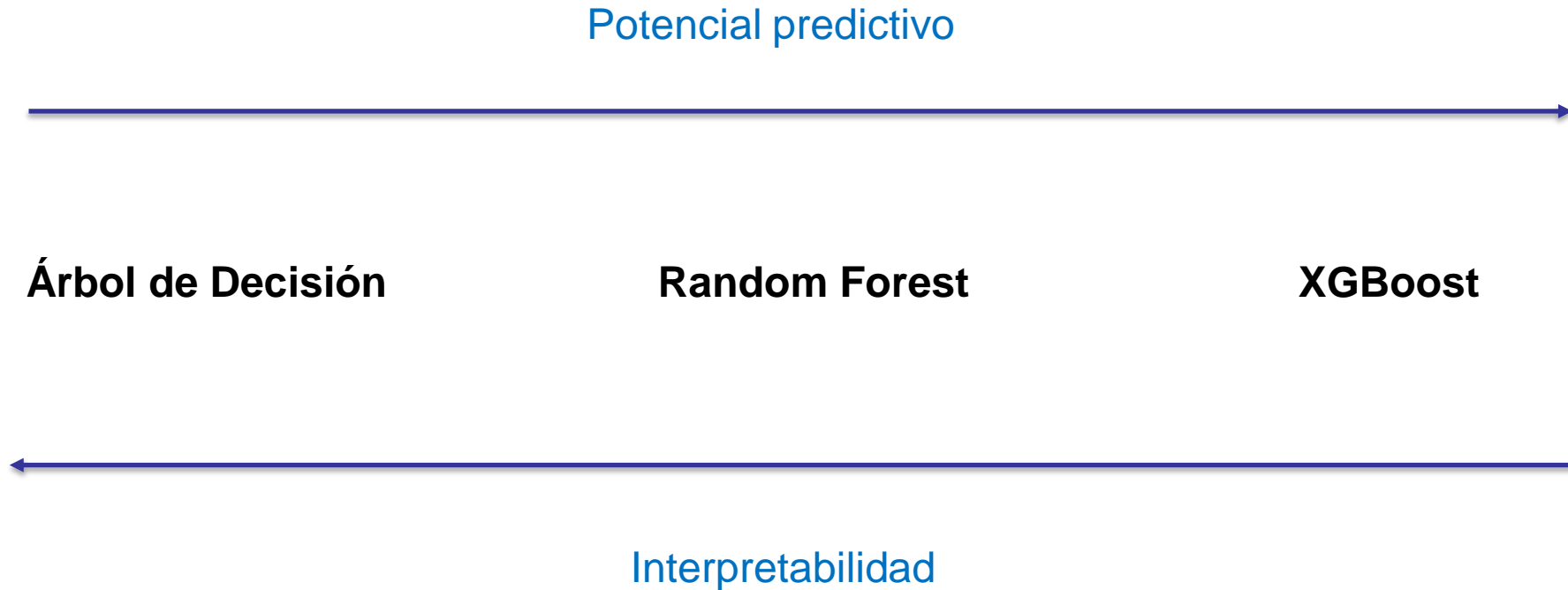
early_stopping_rounds

- Numero de iteraciones consecutivas sin mejorar en el error de validación que se permiten antes de dar por terminado el entrenamiento del modelo (early stopping).
- Al terminar el entrenamiento se devuelve el modelo en la iteración con mejor métrica de validación.
- Toma valores entre 1 e infinito.
- Valores más altos implican mayor tiempo de entrenamiento y un posible sobreajuste al conjunto de validación.
- Valor por defecto: None (No se realiza early stopping).

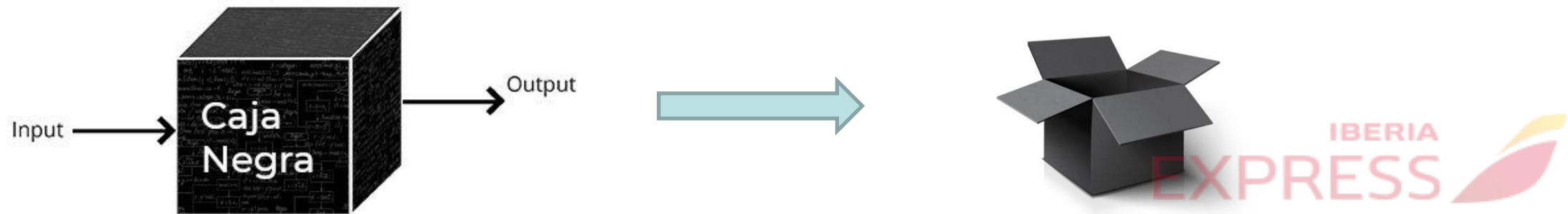
Resumen

Hiperparámetro	Efecto	Rango	Valor por defecto
nrounds	Número de iteraciones de boosting a realizar	$[1, \infty]$	100
eta	Inversamente relacionado con el step size.	$[0, 1]$	0,3
gamma	Mínima mejora en la loss function requerida.	$[0, \infty]$	0
max_depth	Profundidad máxima de cada árbol.	$[1, \infty]$	6
min_child_weight	Suma mínima de pesos necesaria en un nodo hijo.	$[0, \infty]$	1
subsample	Ratio de submuestreo de las instancias de entrenamiento.	$(0, 1]$	1
colsample_bytree	Ratio de submuestreo de las columnas del dataset.	$(0, 1]$	1
num_parallel_tree	Número de árboles calculados en cada iteración.	$[1, \infty]$	1
lambda	Controla la regularización L2.	$[0, \infty]$	1
alpha	Controla la regularización L1.	$[0, \infty]$	0
early_stopping_rounds	Numero de iteraciones sin mejora permitidas.	$[1, \infty]$	None

INTERPRETABILIDAD VS PRECISION (I)



- En aprendizaje automático habitualmente los **modelos más precisos son los más complejos**.
- Por esta razón, los modelos que dan mejores resultados predictivos son normalmente los **menos interpretables**.
- Hasta ahora, había que tomar una decisión entre un **modelo interpretable** o un **modelo que funcionara bien** a nivel predictivo.
- Sin embargo, este **enfoque ha empezado a cambiar** recientemente.



- Algunos factores que han influido en el cambio de enfoque son:
 - Nuevas leyes de protección de datos, **GDPR**.
 - Uso en **sanidad**.
- La extracción de interpretabilidad es una línea de investigación popular en estos momentos, pero aún es un problema sin resolver por completo.
- Sin embargo, ya se han desarrollado algunos algoritmos:
 - **Arboles de decisión** para explicar las decisiones del modelo.
 - Predicción basada en **ensembles de reglas**.
 - **Métodos estadísticos** basados en insertar ruido a los inputs y analizar variaciones en el output.

- **ID Paciente: 0000001**
 - **Predicción de probabilidad de Exitus: 0.9997221**
-

1. des_d1: Infarto agudo de miocardio con elevación de ST (IAMCEST) (IMEST) (STEMI) con implicación de otra arteria coronaria de cara inferior
2. des_grd: OXIGENACIÓN POR MEMBRANA EXTRACORPÓREA (ECMO)
3. des_d2: Insuficiencia cardiaca, no especificada
4. flag_uci: 1
5. des_p1: Derivación de mamaria interna, izquierda a arteria coronaria, una arteria, abordaje abierto
6. edad: 78
7. des_grd_ent: OXIGENACIÓN POR MEMBRANA EXTRACORPÓREA (ECMO)
8. des_d7: Bloqueo auriculoventricular, completo
9. numproc: 15
10. previous_diag: 14

Ventajas

- Una de las familias de modelos de mayor **potencial predictivo**.
- Cada **modelo individual** es muy **poco costoso** computacionalmente.
- **Poco** gasto de **memoria**.
- El elevado número de hiperparámetros permite su **ajuste a problemas muy variados**.
- La implementación oficial está **paralelizada** internamente.

Desventajas

- Modelo complejo → **Poco interpretable de forma directa**.
- **Muchos hiperparámetros** a probar, por lo que las búsquedas en rejilla pueden ser extensas y costosas.
- **Sensible** a la elección de **hiperparámetros**.
- **No** proporciona una **probabilidad calibrada**, solo un score.

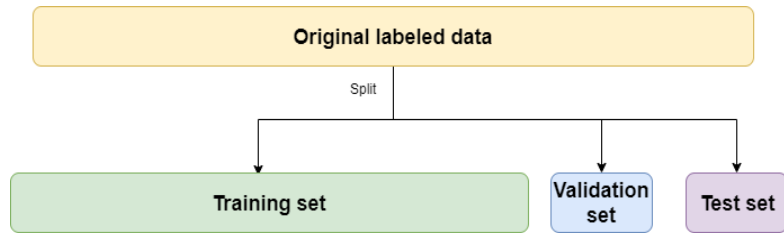
RESUMEN FINAL



Definición Machine Learning

A machine learns with respect to a particular task T , performance metric P , and type of experience E , if the system reliably improves its performance P at task T , following experience E

Cross Validation y validación fija



Split 1	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Metric 1
Split 2	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Metric 2
Split 3	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Metric 3
Split 4	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Metric 4
Split 5	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Metric 5

Below the table, there are two boxes: a red box labeled 'Training data' and a blue box labeled 'Val data'.

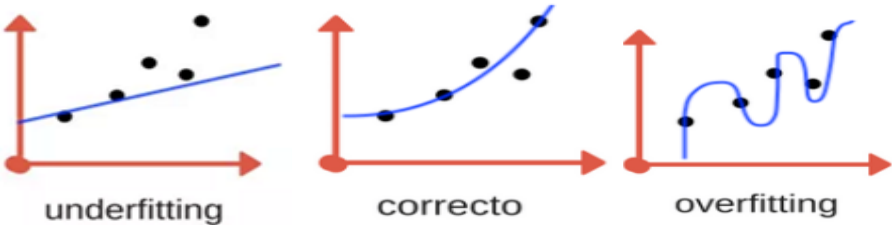
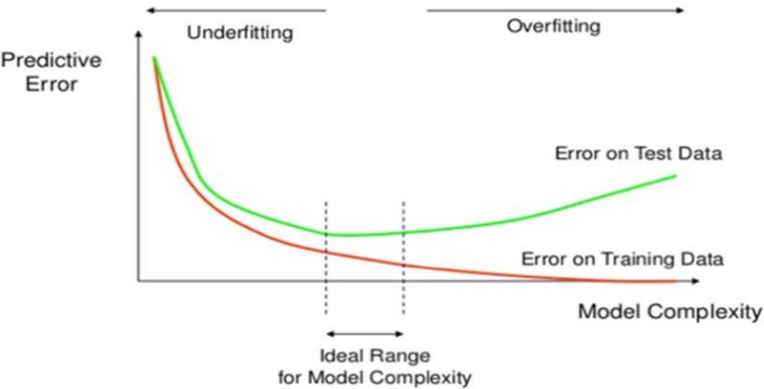
- TRAINING : Datos de los que los modelos extraen patrones.
- VALIDATION : Se emplea para seleccionar el mejor de los modelos entrenados en metamodelización.
- TEST : Proporciona el error real esperado con el modelo seleccionado.

Grid Search

- Los modelos ML suelen incluir un conjunto de **hiperparámetros** que nos permiten controlar su comportamiento.
- De su correcta elección dependerá la bondad del modelo entrenado.

par1/par2	10	100	1000
0.1	0.3	0.22	0.25
0.01	0.15	0.14	0.14
0.001	0.35	0.05	0.11

Overfitting y Underfitting



¿Cómo detectar el overfitting?

Validación tiene un error mucho mayor que en train

¿Cómo detectar el underfitting?

El error de train parece demasiado elevado o da la misma respuesta siempre.

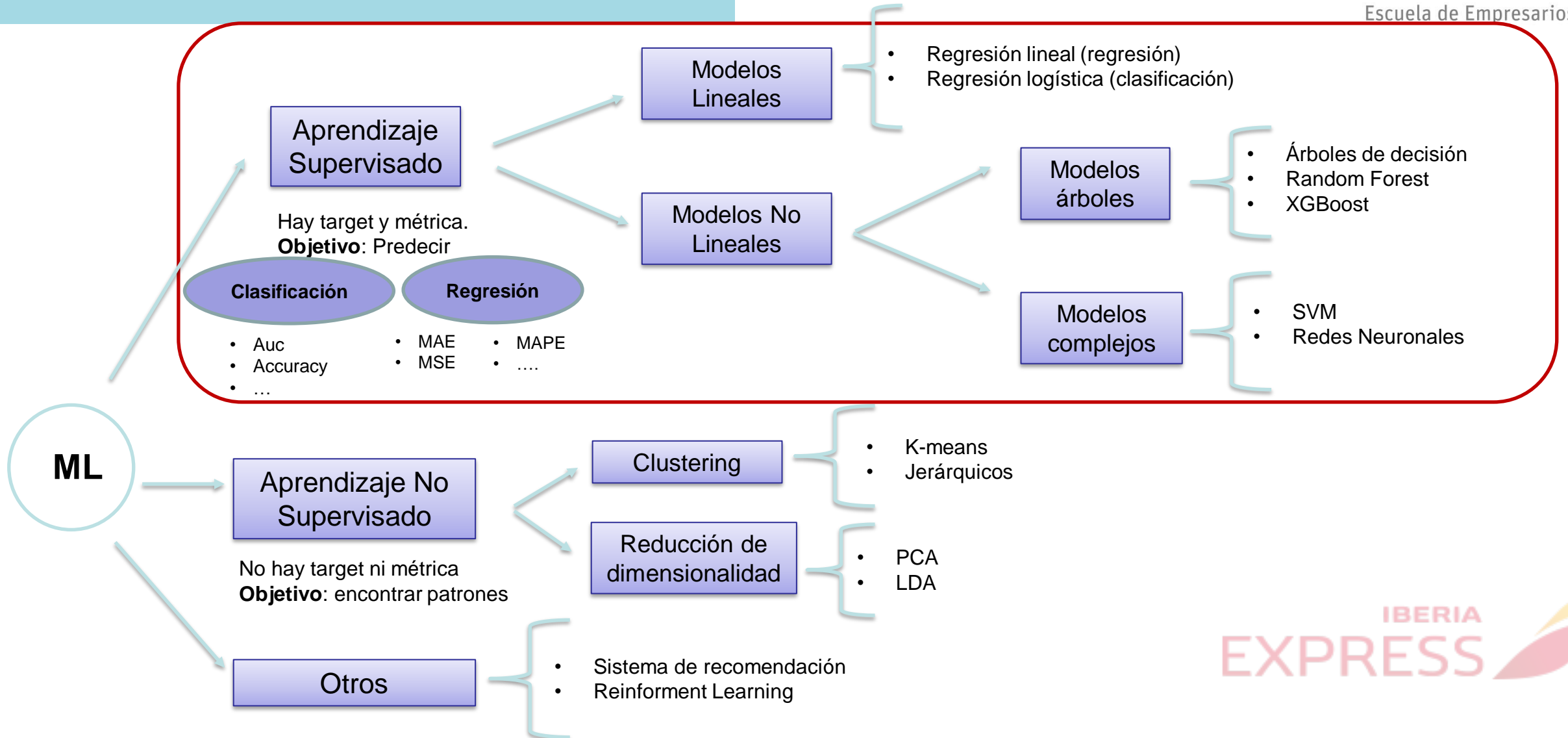
Supervisado VS no Supervisado

	Supervisado	No supervisado
Etiquetas	SI	NO
Objetivo	Dar predicciones a futuro sobre el conjunto de test	Encontrar patrones en los datos o reducir dimensiones
Modelos	Regresión lineal, árboles, SVM, Redes Neuronales	Clustering, PCA
Ejemplo	Predecir si una transacción es fraudulenta	Encontrar clientes con perfiles similares

Clasificación VS Regresión

	Clasificación	Regresión
Etiquetas	Categóricas.	Numéricas.
Ejemplo	Una imagen es un gato (1) o no (0). 	Precio de alquiler de una casa 
Métrica	AUC	MSE

RESUMEN MODELOS



RESUMEN MODELOS

Modelo	Alcance	Definición	Ventaja	Desventaja	Hiperparámetros
Regresión lineal	Regresión	Ajustar una recta lo mejor posible a un grupo de elementos	El más sencillo e interpretable.	Sencillo. No capta relaciones no lineales.	<ul style="list-style-type: none"> • Lasso o Ridge. • alpha (regularización).
Regresión logística	Clasificación	Estima probabilidades utilizando una función logística al resultado de una regresión lineal.	Opción más sencilla para clasificación.	Demasiado sencillo para la mayoría de problemas.	<ul style="list-style-type: none"> • Lasso o Ridge. • alpha (regularización).
SVM	Ambos	Encontrar el hiperplano de máximo margen o ajuste.	Uno de los modelos más potentes. Solución asegurada y única.	Entrenamiento costoso. Sensible a hiperparámetros.	<ul style="list-style-type: none"> • C • γ • ϵ
Árbol de decisión	Ambos	Cada nodo es una variable, cada rama una decisión y cada nodo hoja un output.	Interpretable. Admite variables categóricas.	Demasiado sencillo para la mayoría de problemas.	max depth, min samples split, min samples leaf, max features.
Random Forest	Ambos	Combinación de árboles de decisión con voto.	Proporciona variable importance. Robusto a hiperparámetros.	Menor potencial predictivo que SVM, NN o XGB.	<ul style="list-style-type: none"> • ntree • mtry • nodesize
XGBoost	Ambos	Combinación iterativa de árboles o RF que corrigen errores de los anteriores.	Gran potencial predictivo. Muy rápido.	Muchos hiperparámetros que optimizar.	nrounds, eta, gamma, max depth, min child weight, subsample, colsample bytree, num parallel tree, lambda, alpha, early_stopping_rounds.

Yvonne Gala García:
yvonne.gala@iberiaexpress.com

Jesús Prada Alonso:
ext.jprada@iberiaexpress.com