

MACHINE LEARNING INMERSION

ANDRÉ OMAR CHÁVEZ PANDURO

« El mayor enemigo del conocimiento no es la ignorancia, es la ilusión del conocimiento »



AGENDA

- Clasificación.
- Modelo General de los Métodos de Clasificación.
- Sesgo vs Varianza - Bias vs Variance.
- Modelos de Boosting.
- Bagging vs Boosting.
- AdaBoost.
- GBM.
- XgBoost.
- LigthGBM.
- CatBoost.



CLASIFICACIÓN: DEFINICIÓN

- Dada una colección de registros (Conjunto de Entrenamiento) cada registro contiene un conjunto de variables (atributos) denominado x , con una variable (atributo) adicional que es la clase denominada y .
- El objetivo de la **clasificación** es encontrar un modelo (una función) para predecir la clase a la que pertenecería cada registro, esta asignación una clase se debe hacer con la mayor precisión posible.
- Un conjunto de prueba (tabla de testing) se utiliza para determinar la precisión del modelo. Por lo general, el conjunto de datos dado se divide en dos conjuntos al azar de el de entrenamiento y el de prueba.

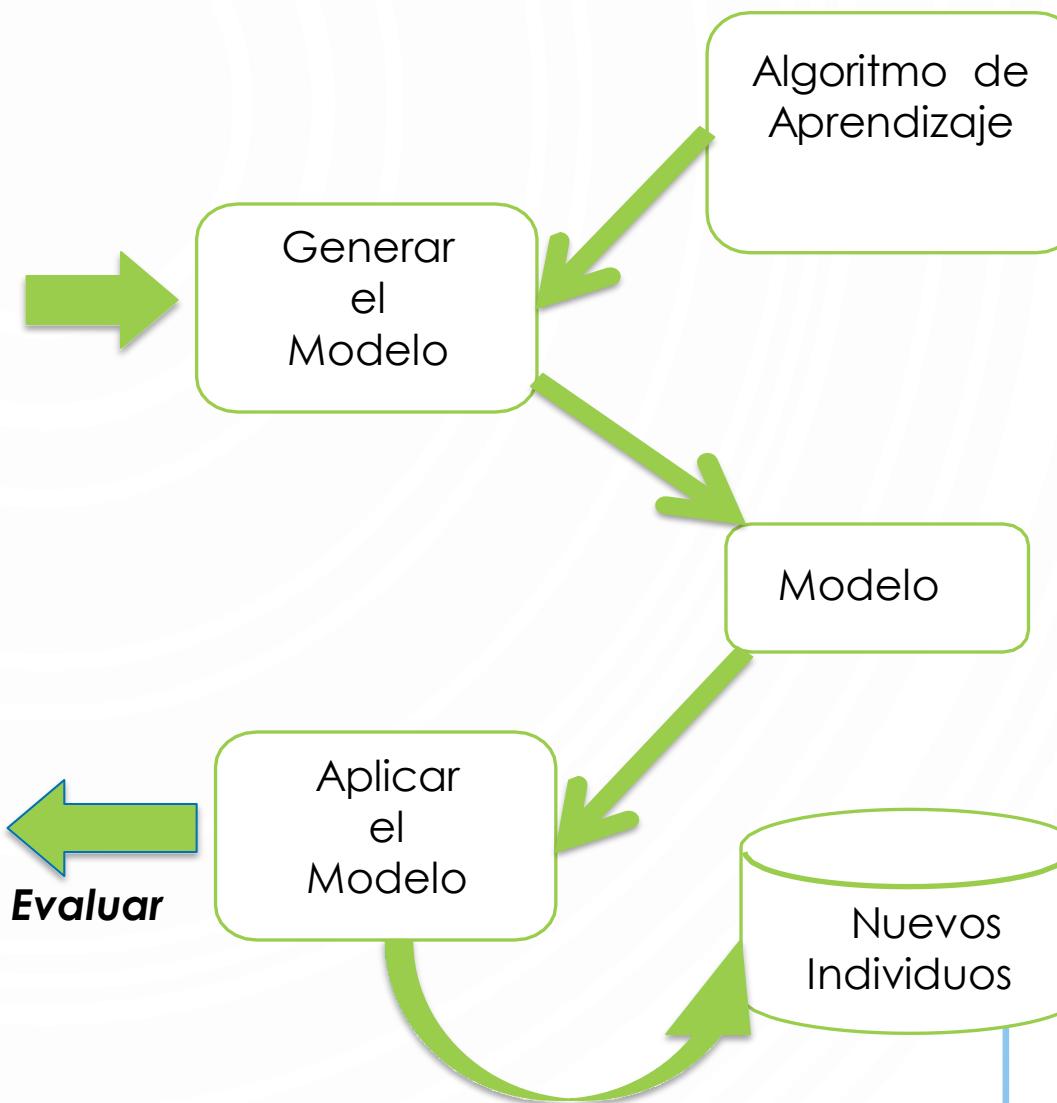
MODELO GENERAL DE LOS MÉTODOS DE CLASIFICACIÓN

ID	REEMBOLSO	ESTADO CIVIL	INGRESOS ANUALES	FRAUDE
1	SI	SOLTERO	S/ 1,000	NO
2	SI	CASADO	S/ 5,000	NO
3	NO	CASADO	S/ 3,500	SI
4	SI	VIUDO	S/ 4,500	NO
5	NO	SOLTERO	S/ 2,000	NO
6	NO	SOLTERO	S/ 1,500	SI

Tabla de Aprendizaje

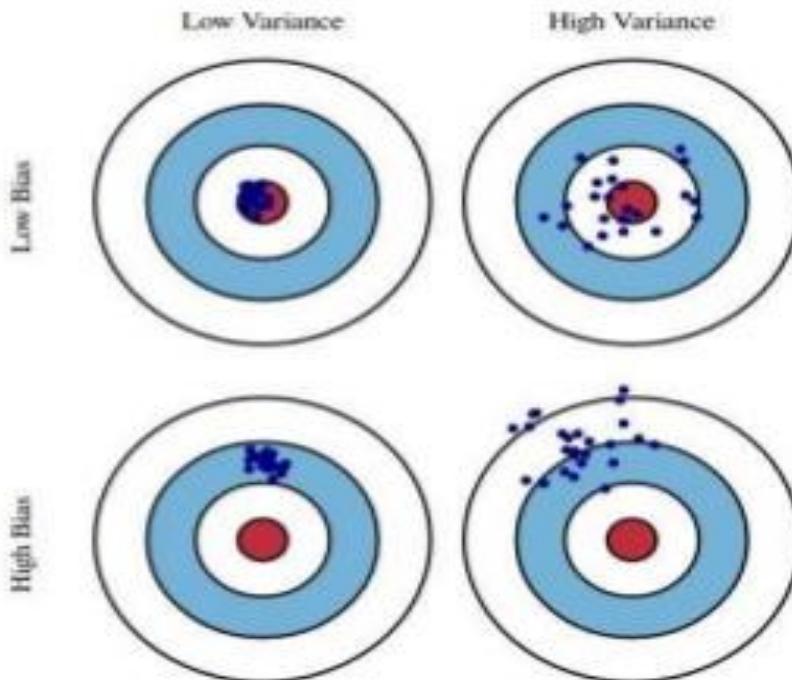
ID	REEMBOLSO	ESTADO CIVIL	INGRESOS ANUALES	FRAUDE
7	SI	SOLTERO	S/ 4,000	NO
8	SI	CASADO	S/ 5,500	NO
9	NO	CASADO	S/ 6,500	SI

Tabla de Testing



EL PROBLEMA DE BIAS Y VARIANZA

Errores y Medidas



Trade off :

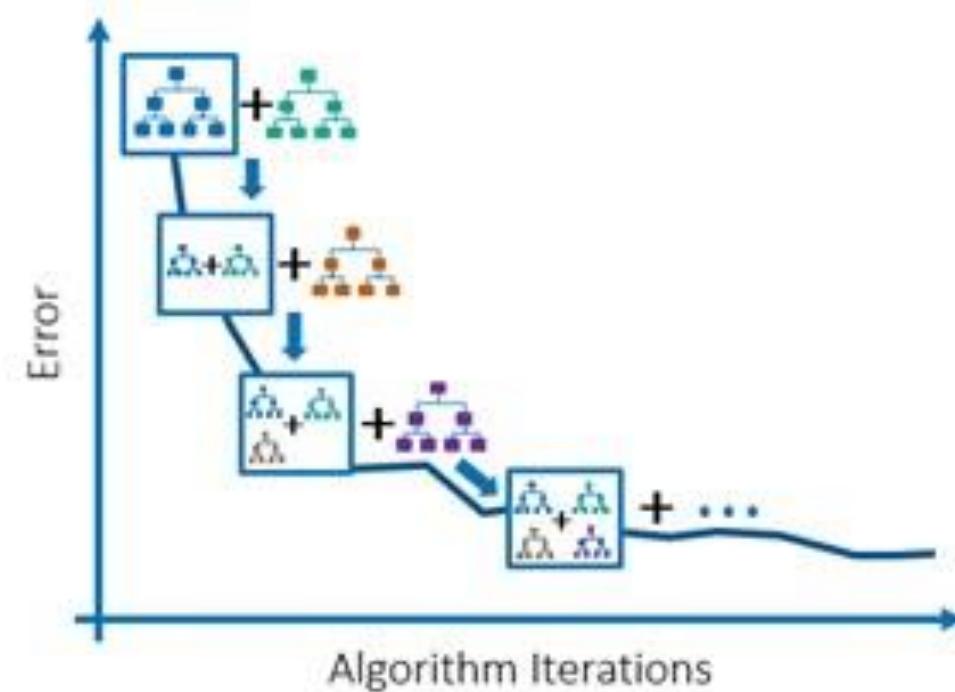
- Over-Fitting :
 - Bias bajo, Varianza alta.
 - Modelo complejo y flexible.
- Under-Fitting :
 - Bias alto, Varianza baja.
 - Modelo simple y rígido.

Normalmente:

$$\text{Bias } \downarrow \downarrow \Rightarrow \text{Varianza } \uparrow \uparrow$$

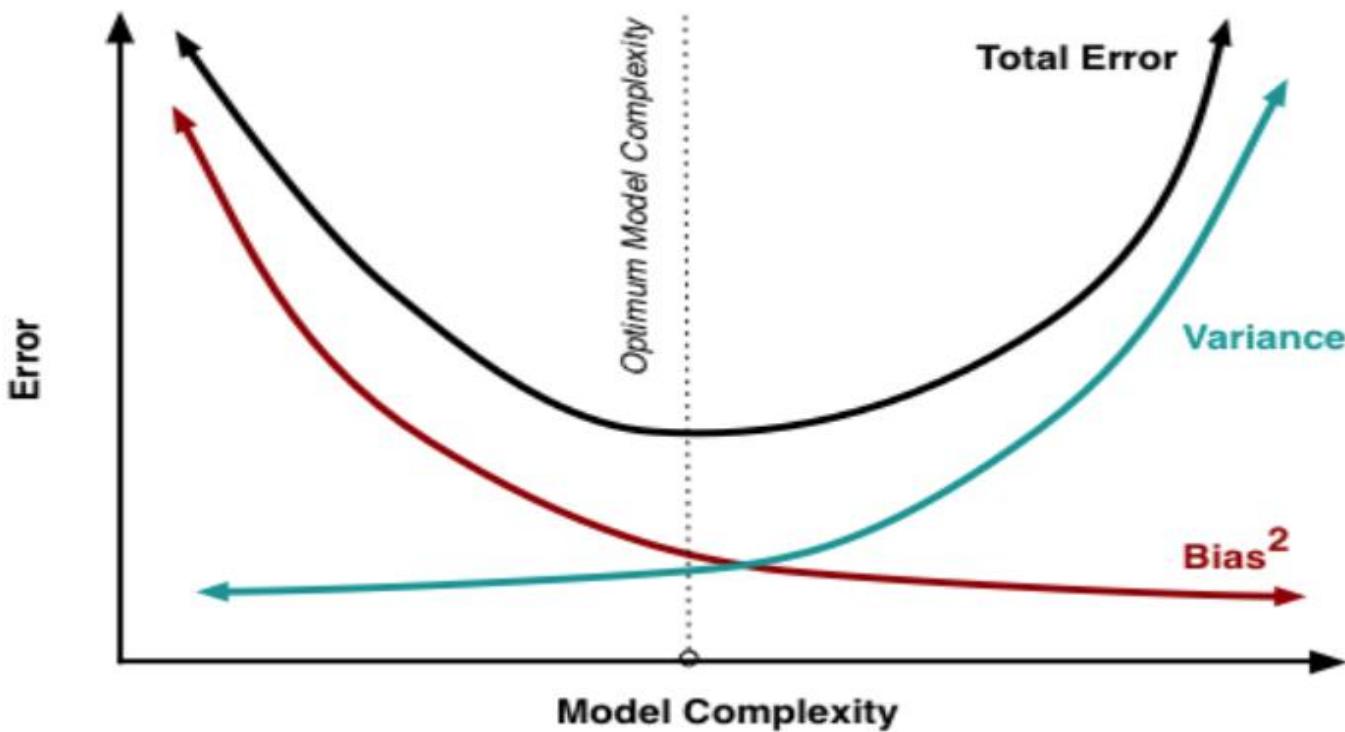
$$\text{Varianza } \downarrow \downarrow \Rightarrow \text{Bias } \uparrow \uparrow$$

LA GANANCIA EN LA COMPLEJIDAD DE LOS ALGORITMOS *



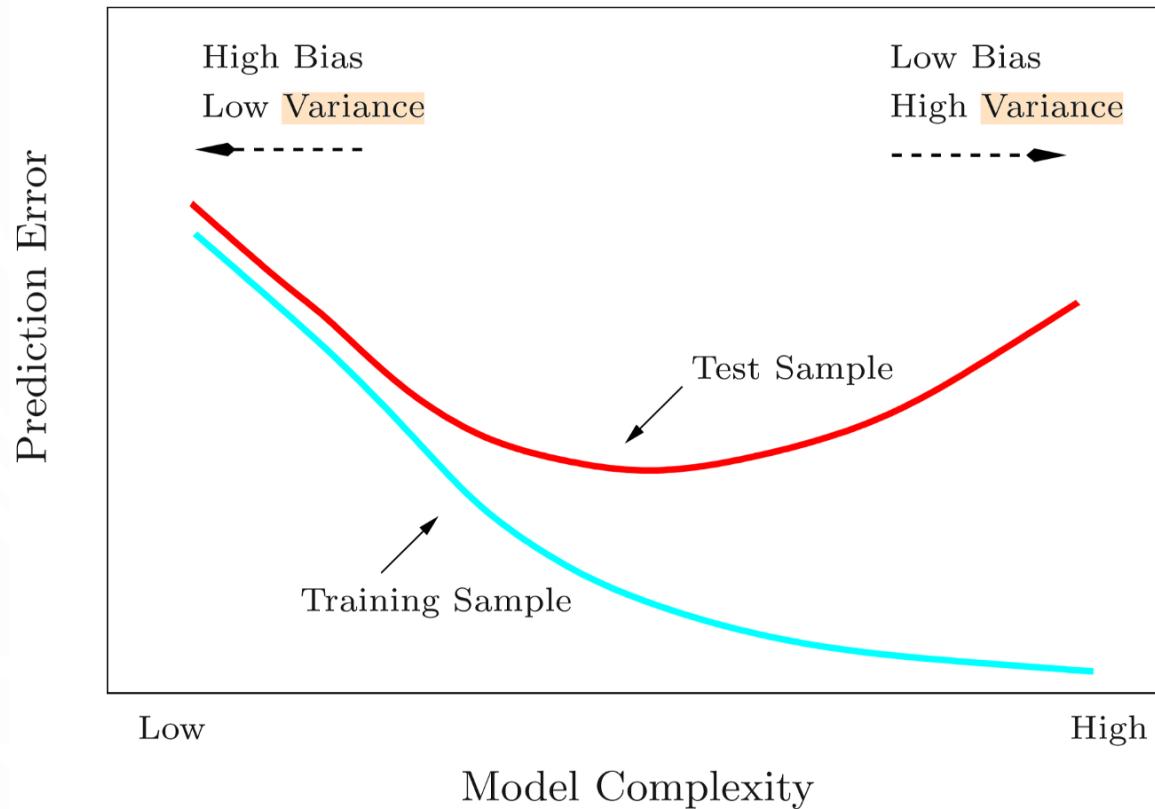
EL PROBLEMA DE BIAS Y VARIANZA

¿Cuál es el mejor modelo?



EL PROBLEMA DE BIAS Y VARIANZA

¿Qué sucede en entrenamiento y validación?



MÉTODOS DE ENSAMBLE : IDEA INTUITIVA Y NEURONAL

➤ La clave para que los métodos de **ensemble** consigan mejores resultados que cualquiera de sus modelos individuales es que, los modelos que los forman, sean lo más diversos posibles.

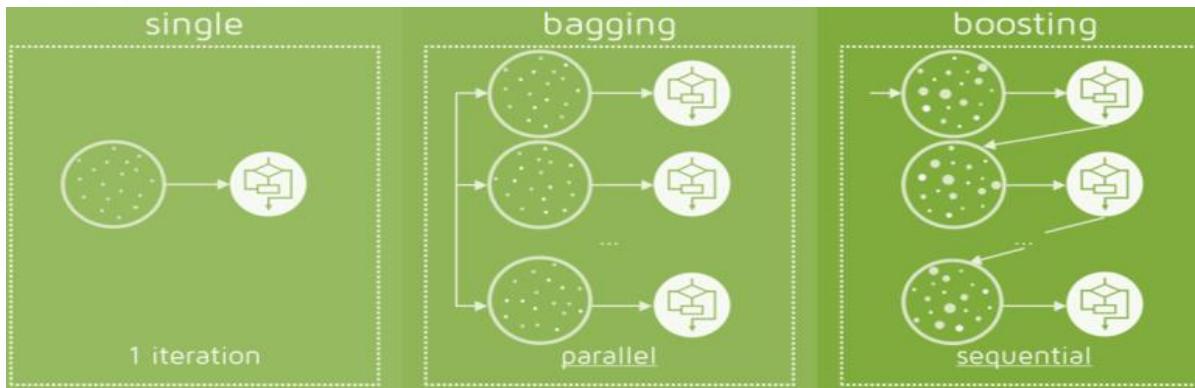
JUEGO DE PREGUNTAS SOBRE TEMÁTICAS DIVERSAS



MÉTODOS DE ENSAMBLE : DIFERENCIAS IMPORTANTES

- La estrategia seguida para reducir el error total: El error total de un modelo puede descomponerse como **bias+varianza+ ϵ** .

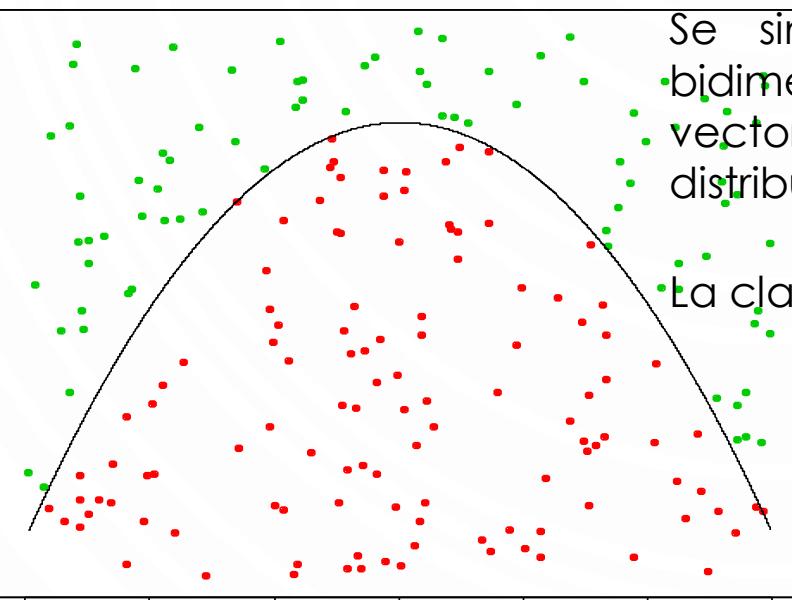
- ✓ En **bagging**, se emplean modelos con muy poco **bias** pero mucha **varianza**, agregando muchos de estos modelos se consigue reducir la **varianza** sin apenas inflar el **bias**.
- ✓ En **boosting**, se emplean modelos con muy poca varianza pero mucho **bias**, ajustando secuencialmente muchos modelos se reduce el **bias**.



EJEMPLO DE RESOLUCIÓN INTUITIVO

Problema de clasificación binaria. Frontera de decisión cuadrática

Frontera de decisión Bayes. Tasa error = 0



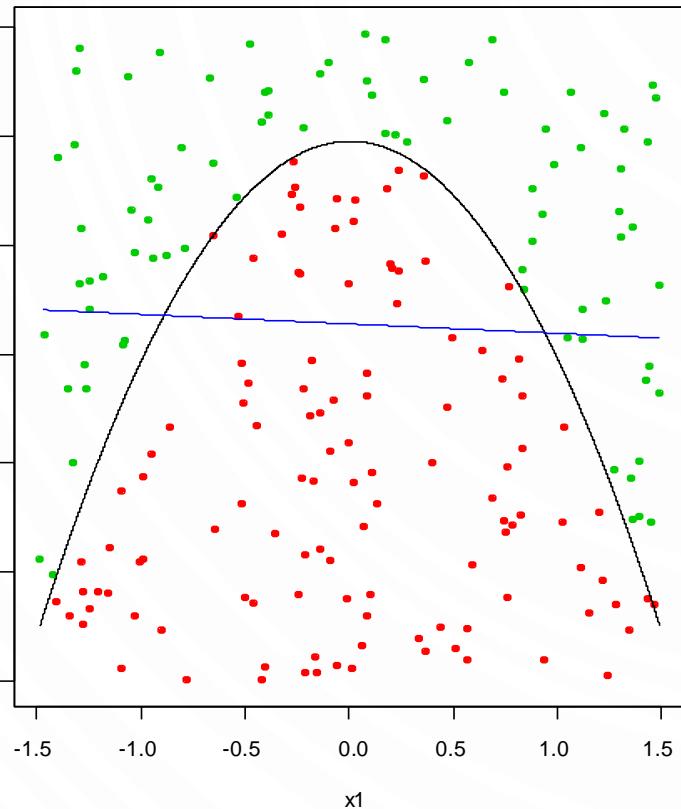
Se simularon 200 observaciones del vector bidimensional (x_1, x_2) ; ambas componentes del vector son variables independientes con una distribución uniforme en $(-1.5, 1.5)$

La clasificación viene dada por:

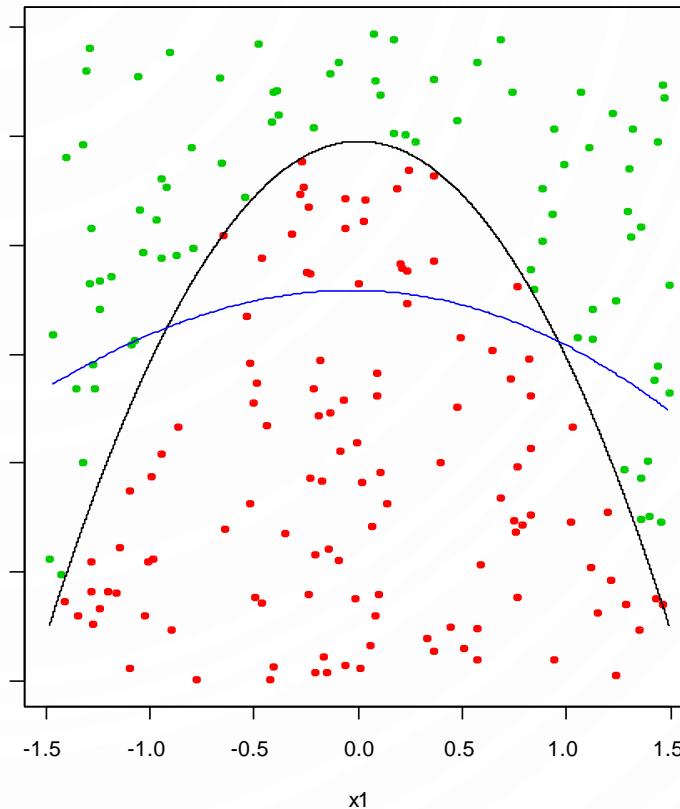
$$\text{Clase 2: } x_2 > 1 - x_1^2 \quad \text{Clase 1: } x_2 < 1 - x_1^2$$

SOLUCIONES LDA Y QDA

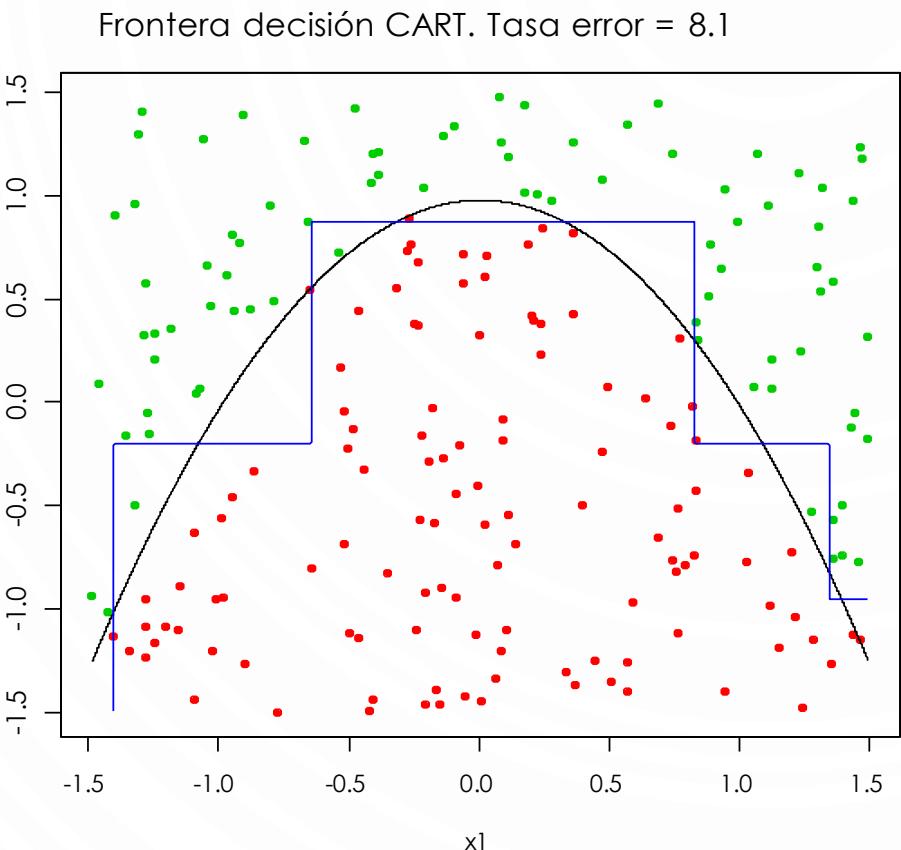
Frontera decisión LDA. Tasa error = 19.91



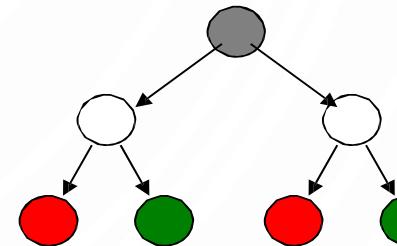
Frontera decisión QDA. Tasa error = 15.75



SOLUCIÓN ÁRBOL DE DECISIÓN

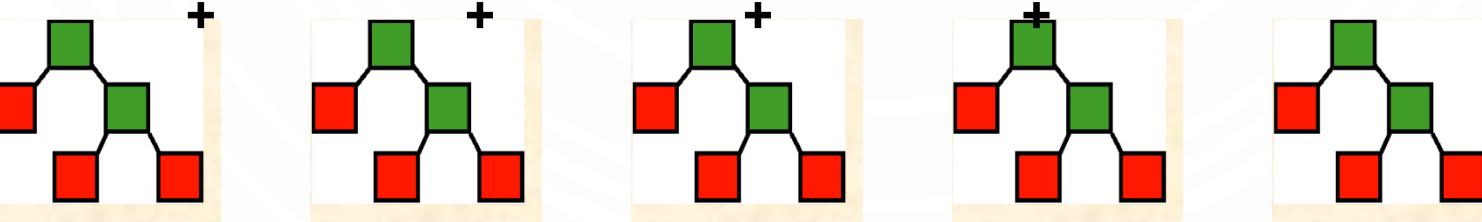


Solución obtenida mediante Árbol sin podar **CART.**



¿QUÉ ES BAGGING?

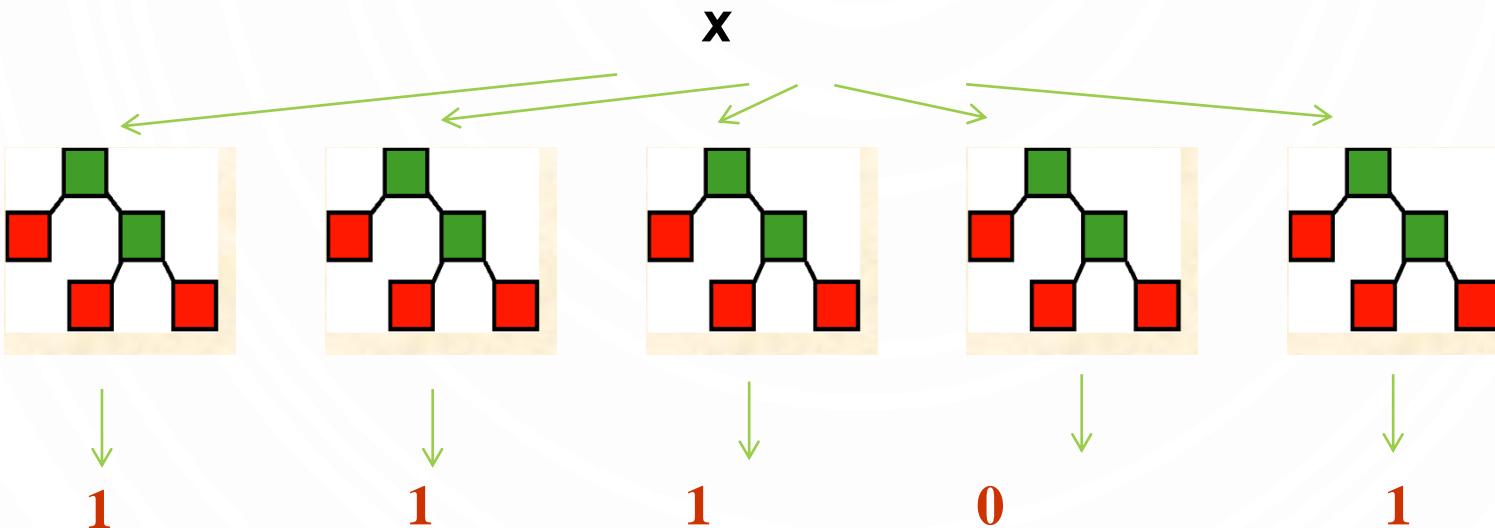
- Bagging quiere decir bootstrap aggregation. Introducido por Leo Breiman (Berkeley) en 1996
- La idea es simple. Si tienes las opiniones de un comité de expertos, considéralas todas para tomar una decisión
- Se extraen muestras bootstrap del conjunto de datos. Para cada muestra, se obtiene un modelo de predicción. El nuevo predictor “bagging” se construye mediante agregación



➤ El objetivo es reducir la inestabilidad

BAGGING PARA CLASIFICACIÓN

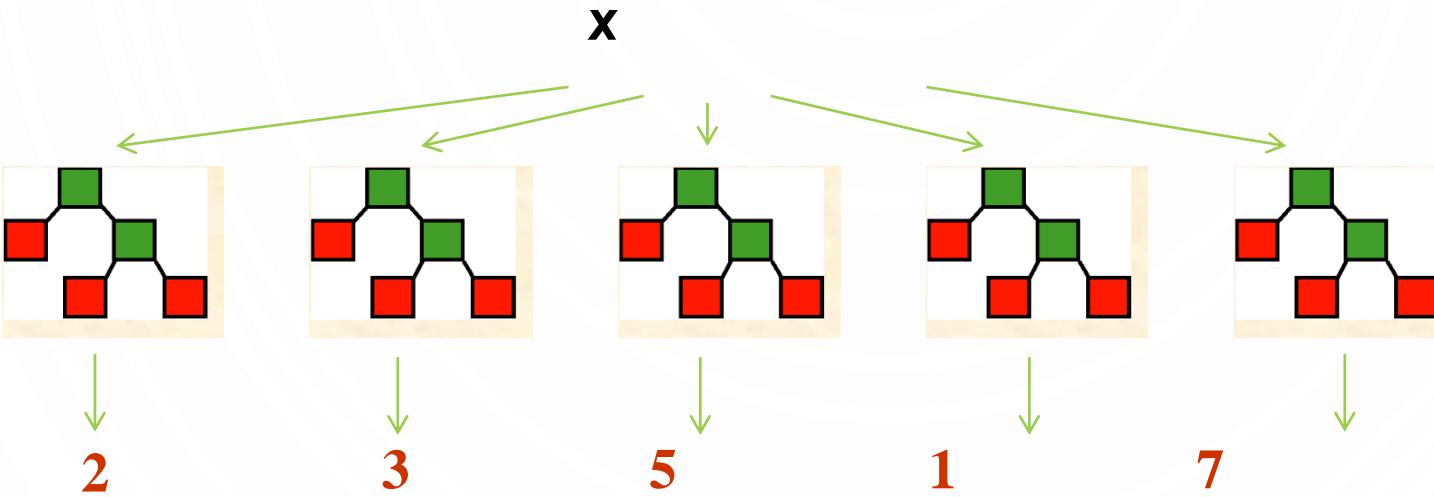
- Si el problema es de **claseficación** bagging clasificará cada nueva observación por mayoría.
- Por ejemplo:



- La clase 1 recibió cuatro votos. La clase 0 un voto.
- El predictor bagging clasificará x en la clase 1.

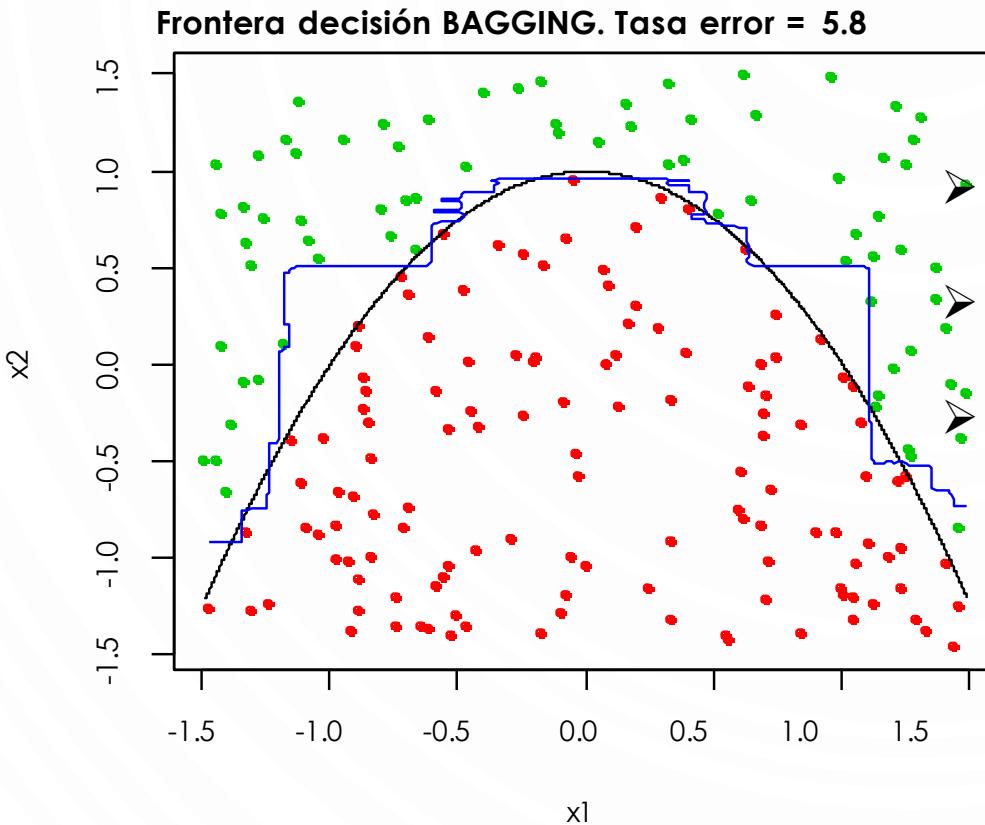
BAGGING PARA REGRESIÓN

- Si el problema es de **regresión** la predicción bagging se obtiene promediando las predicciones de todos los modelos. Por ejemplo:



- La predicción bagging será: $(2+3+5+1+7)/5 = 3.6$
- Cuando la variable respuesta es binaria 0/1, el bagging para regresión se reduce al criterio de clasificar por mayoría.

EJEMPLO 4: EL EFECTO BAGGING



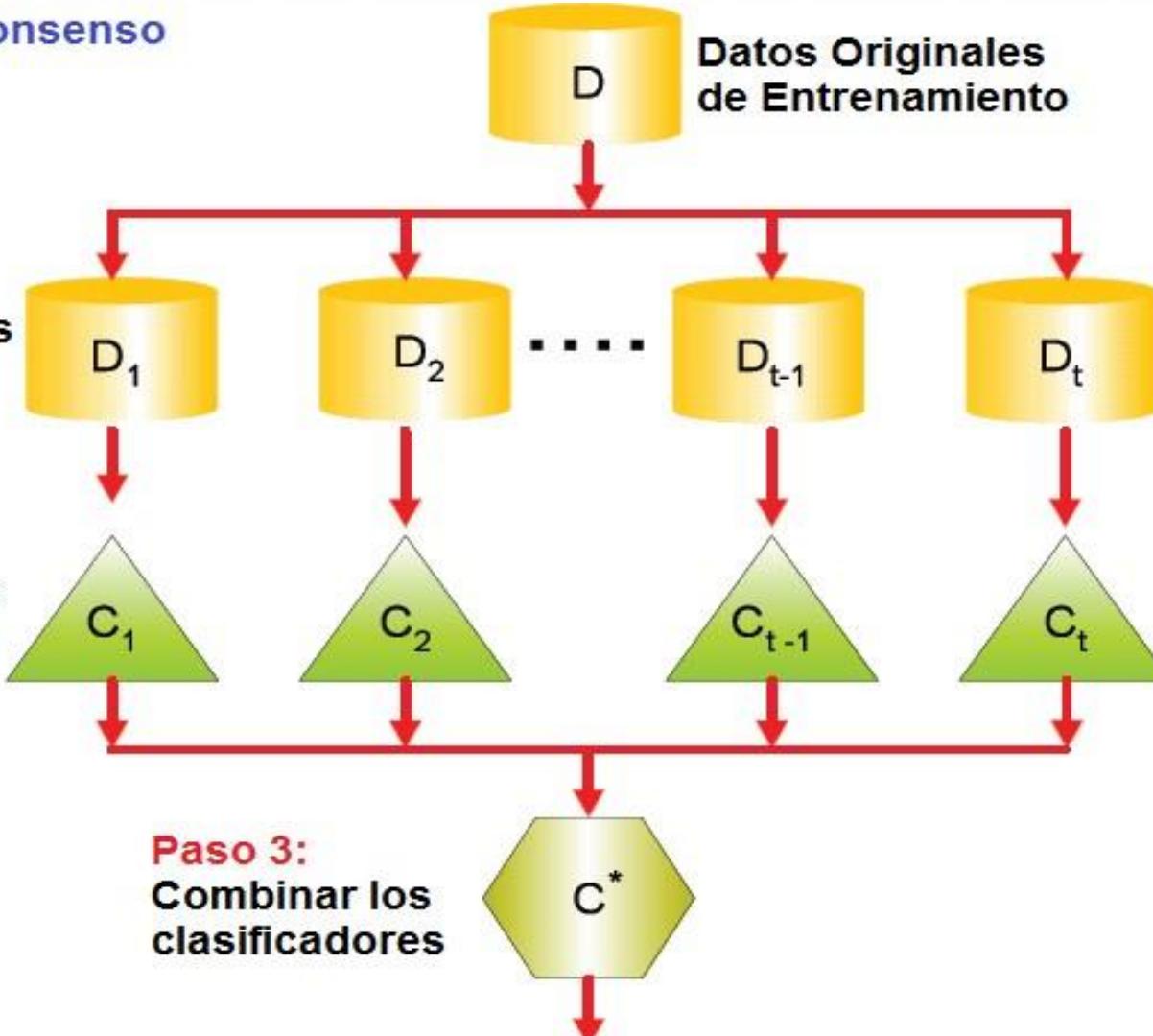
Se emplearon 50 muestras bootstrap.
Se ha reducido la inestabilidad de CART.
En este caso también se ha mejorado en capacidad predictiva.

Métodos de Consenso

Paso 1:
Crear múltiples
conjuntos de
datos

Paso 2:
Crear múltiples
clasificadores

Datos Originales
de Entrenamiento



Paso 3:
Combinar los
clasificadores

BOSQUES ALEATORIOS (RANDOM FOREST)

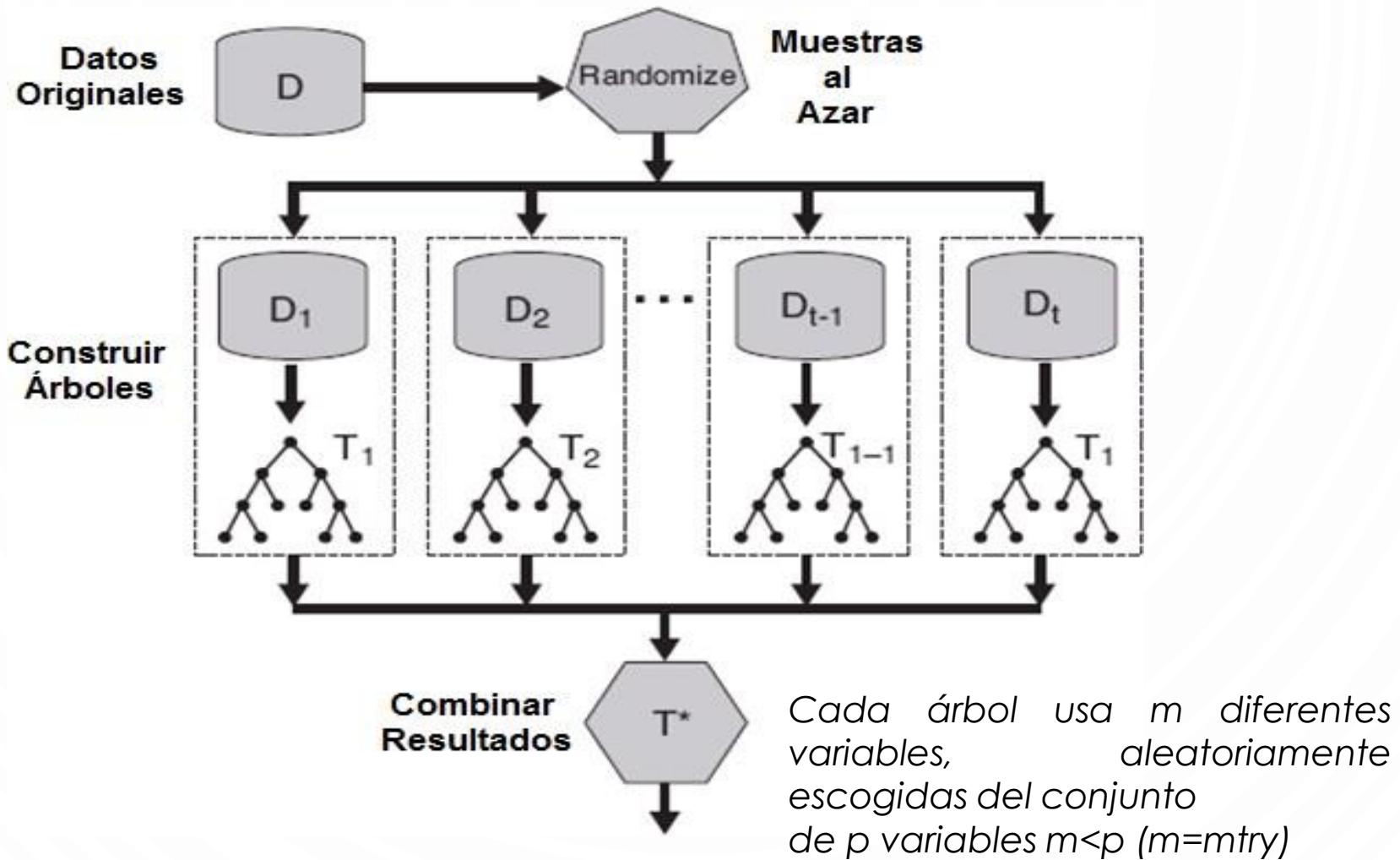
- El caso en el que todos los clasificadores del Método de Consenso son Árboles dicho método se denomina Bosques Aleatorios (Random Forest).



RANDOM FORESTS O BOSQUES ALEATORIOS (RF)

- Desarrollado por Leo Breiman (Berkeley) en 2001
- Tiene su base en
 - ✓ La predicción con CART
 - ✓ La agregación de modelos de árbol
 - ✓ Bootstrap Aggregation (Bagging)
- Comercializado por Salford Systems en la herramienta **RandomForests™**.
- Implementado por Andy Liaw y Matthew Wiener en la librería **randomForest** del entorno R de programación.

BOSQUES ALEATORIOS (RANDOM FOREST)



EL MECANISMO DE RANDOM FOREST

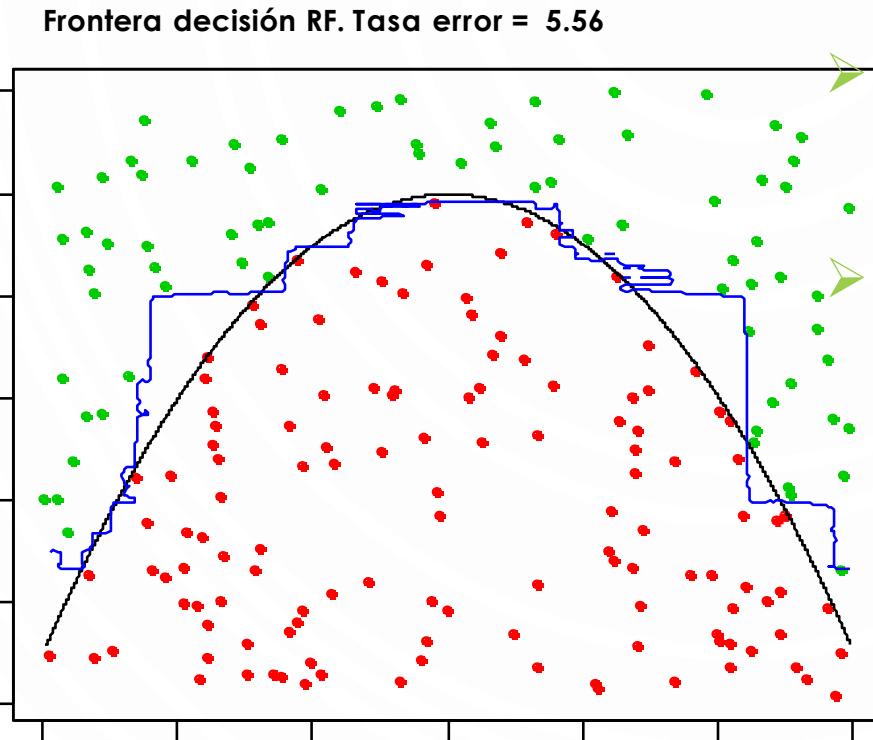


- La aleatorización se introduce en el mecanismo de aprendizaje a través de dos vías: el **remuestreo** y la **aleatorización en la selección del corte** en cada nodo
- Se toman B muestras bootstrap del conjunto de datos para construir B árboles sin podar. Esto corresponde a la fase bagging y proporciona el bosque de árboles
- Para construir cada árbol del bosque, RF busca el corte en cada nodo entre un conjunto de R variables predictoras que han sido seleccionadas al azar
- Por defecto $B = 500$ y $R = \text{sqrt}(n^{\circ} \text{ predictores})$.

LA ESTIMACIÓN DEL ERROR CON RF

- Se define la tasa de error **out of bag** (OOB_i) de una observación x_i como el error obtenido al ser clasificada por los árboles del bosque construidos sin su intervención.
- La estimación **OOB** del error es el promedio de todos los OOB_i para todas las observaciones del conjunto de datos.
- Es mejor estimador que el error aparente. Parecida a la estimación por validación cruzada.
- La medida se puede extrapolar al problema de regresión describiéndola en términos del ECM.

SOLUCIÓN CON RANDOM FOREST



Se empleó un bosque con **5000** árboles

¿Qué ha ocurrido con la tasa de error? Comparar con lda, qda, CART y bagging

DOCUMENTACIÓN SOBRE RANDOM FOREST

- ✓ Página Web de **Leo Breiman**:
<http://www.stat.berkeley.edu/users/breiman/> Fallecido en julio de 2005
- ✓ Página Web de **Adele Cutler**: <http://www.math.usu.edu/~adele/>
- ✓ Página Web de **Salford Systems**: <http://salford-systems.com/> Versión comercial. White papers y muchas aplicaciones de RF en consultoría

MÉTODOS DE ENSAMBLE : BOOSTING

- Consiste en ajustar secuencialmente múltiples modelos sencillos, llamados **weak learners**, de forma que cada modelo aprende de los errores del anterior.
- Como valor final, al igual que en **bagging**, se toma la media de todas las predicciones (variables continuas) o la clase más frecuente (variables cualitativas).
- Tres de los métodos de **boosting** más empleados son **XGBoost**, **LighGBM** y **CatBoost**.

ADABOOST

AdaBoost

Freund & Shapire

$$h_t = \min_h \underbrace{\sum_i D_t(i) e^{-y_i h(x_i)}}_{Z_t}$$

$$\{\alpha_t, \beta_t\} = \frac{1}{2} \log \left(\frac{W_+}{W_-} \right)$$

$$D_{t+1}(i) = \frac{D_t(i) e^{-y_i h_t(x_i)}}{Z_t}$$

**Final classifier is
linear combination of
weak classifiers**

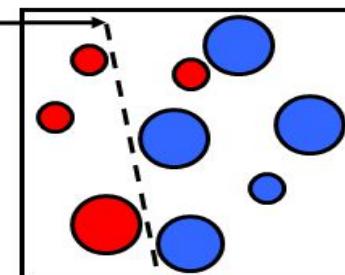
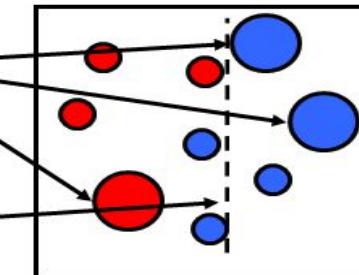
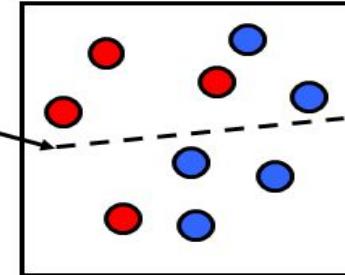
Weak
Classifier 1

Weights
Increased

Weak
Classifier 2

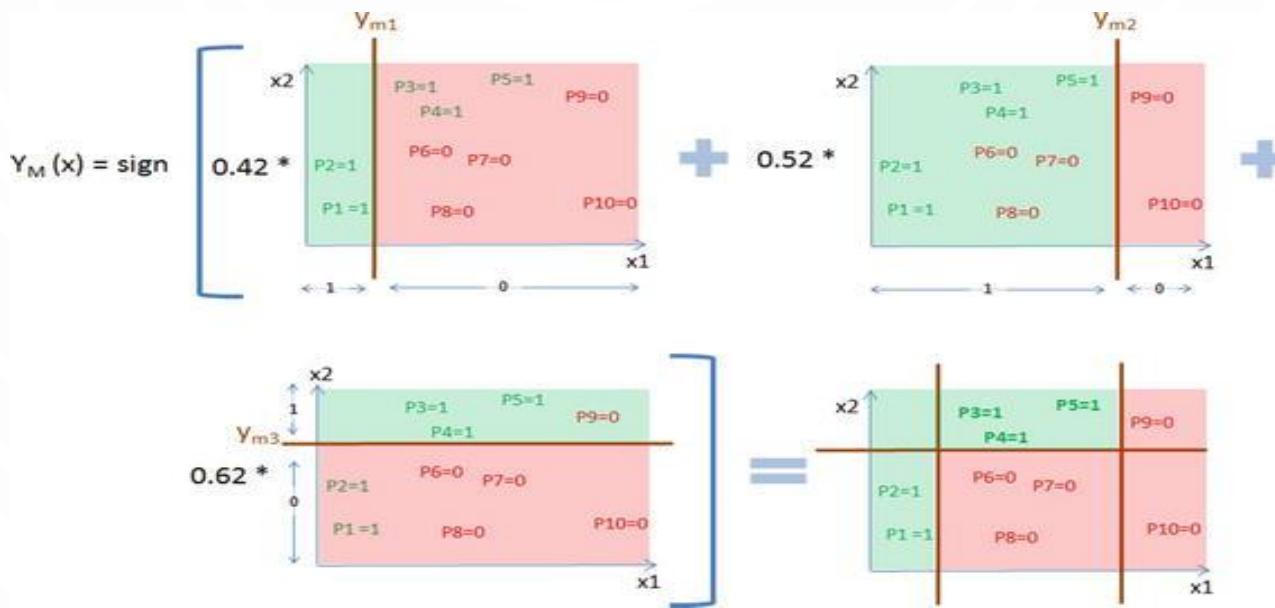
Weak
classifier 3

Viola 2003



ADABOOST

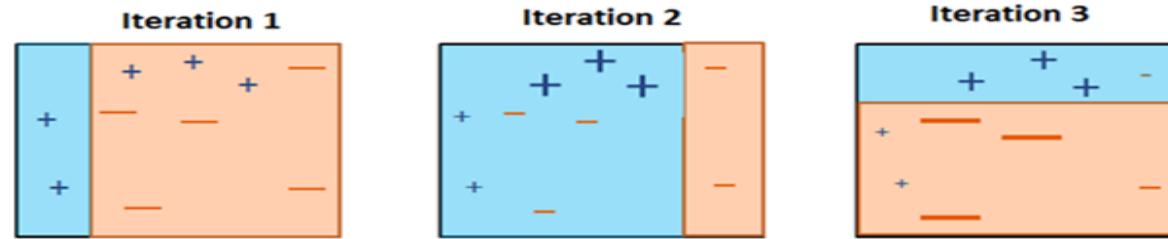
➤ La publicación en 1999 del algoritmo AdaBoost (Adaptive Boosting) por parte de Yoav Freund y Robert Schapire supuso un avance muy importante en el campo del aprendizaje estadístico, ya que hizo posible aplicar la estrategia de boosting a multitud de problemas.



ADABOOST

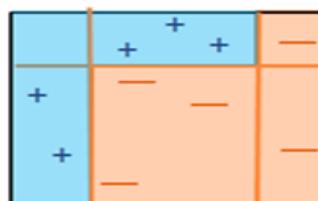
➤ Funcionamiento:

AdaBoost Classifier Working Principle with Decision Stump as a Base Classifier



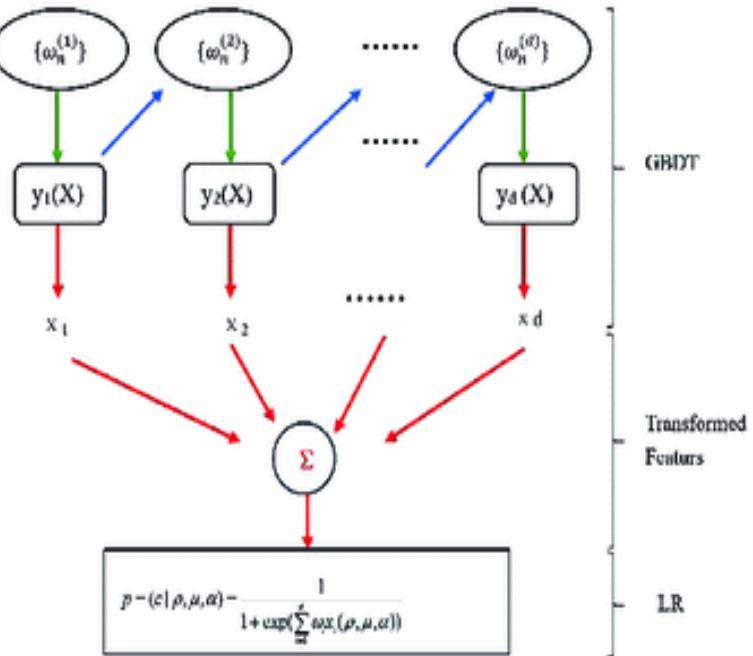
$$H = \text{sign} (0.38 \times \boxed{} + 0.58 \times \boxed{} + 0.87 \times \boxed{})$$

=



Final Classifier / Strong Classifier

GRADIENT BOOSTING



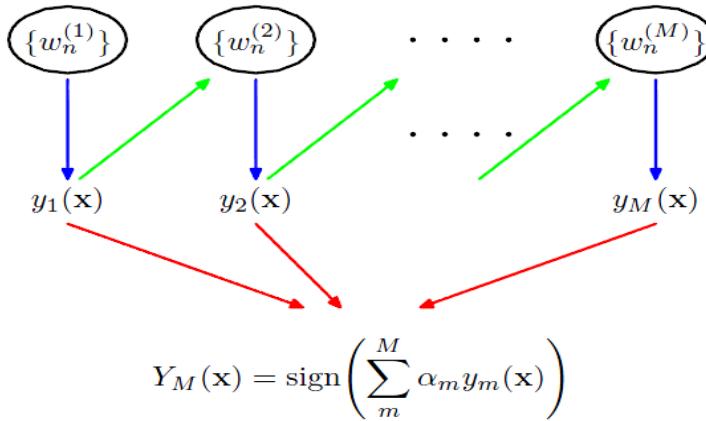
- Gradient Boosting es una generalización del algoritmo AdaBoost que permite emplear cualquier función de coste, siempre que esta sea diferenciable.
- Para cada uno de ellos, el algoritmo de Gradient Boosting es ligeramente distinto, pero, para todos, la idea es la misma: dada una función de coste (por ejemplo, residuos cuadrados para regresión) y un weak learner (por ejemplo, árboles), el algoritmo trata de encontrar el modelo que minimiza la función de coste.

GRADIENT BOOSTING

- Dado que el objetivo de Gradient Boosting es ir minimizando los residuos iteración a iteración, es susceptible de overfitting.
- Una forma de evitar este problema es empleando un valor de regularización, también conocido como **learning rate** (λ), que limite la influencia de cada modelo en el conjunto del **ensemble**.
- Como consecuencia de esta regularización, se necesitan más modelos para formar el **ensemble** pero se consiguen mejores resultados.

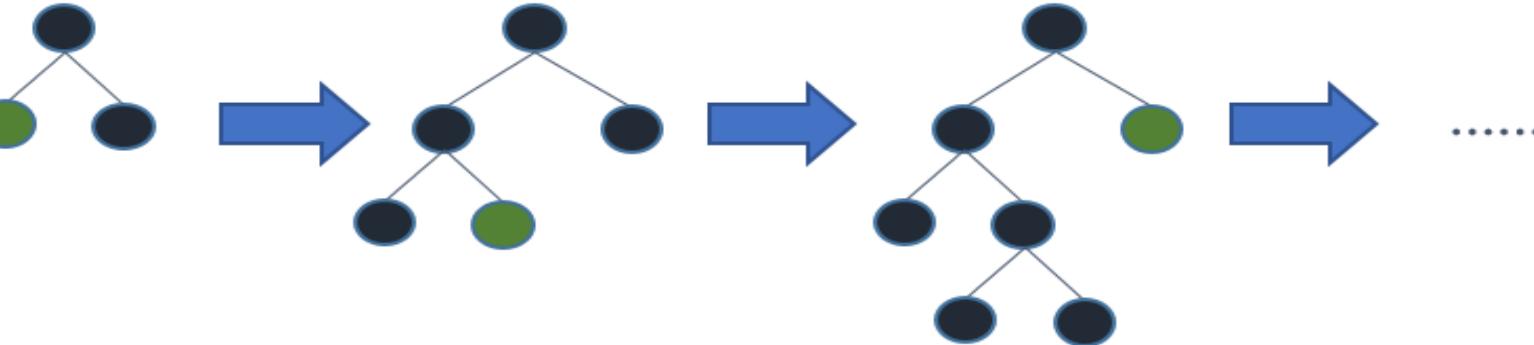
GRADIENT BOOSTING MACHINE

- Gradient Boosting es una generalización del algoritmo AdaBoost que permite emplear cualquier función de coste
- La flexibilidad de este algoritmo ha hecho posible aplicar boosting a multitud de problemas (regresión, clasificación con más de dos clases...).



PARÁMETROS DE GBM (GRADIENT BOOSTING MACHINE)

- **Parámetros específicos del árbol:** Afectan a cada árbol individual en el modelo.
- **Parámetros de refuerzo o boosting:** Afectan la operación de **refuerzo** en el modelo.
- **Parámetros misceláneos:** otros parámetros para el funcionamiento general.



PARÁMETROS DE GBM : PARÁMETROS DEL ARBOL

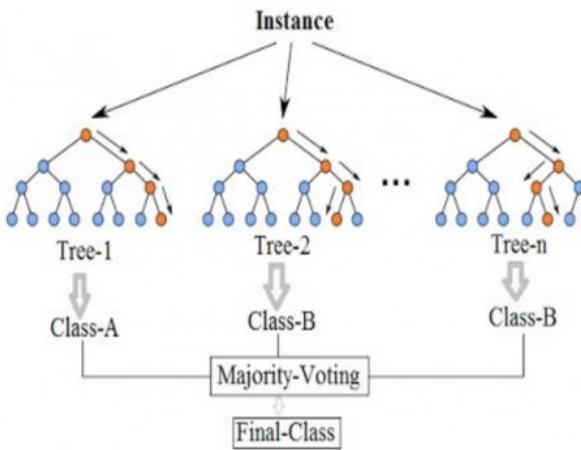
1. **min_samples_split** .- Mínimo número de observaciones para que un nodo se divida.
(Se recomienda valores altos).
2. **min_samples_leaf**
3. **min_weight_fraction_leaf**.- Similar a min_samples_leaf, sólo que se asigna una fracción de observaciones respecto al total.
4. **max_depth**
5. **max_leaf_nodes**.- Número máximo de nodos terminales u hojas por árbol.
6. **max_features**

PARÁMETROS DE GBM : PARÁMETROS DEL BOOSTING

7. **learning_rate**
8. **n_estimators**.- Número de árboles a considerar en el boosting.
9. **Subsample**.- Fracción de observaciones a usar en cada árbol.

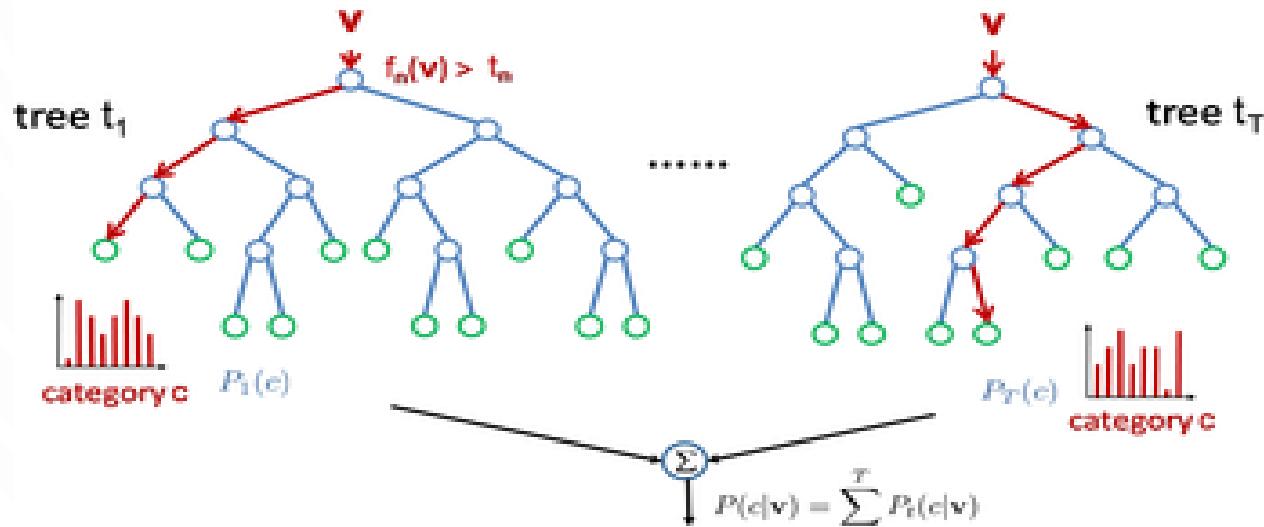
PARÁMETROS DE GBM : PARÁMETROS MISCELÁNEOS

1. **Loss**.- Función de error a ser minimizada en cada corte. Diferentes para regresión o clasificación.
2. **Init**.- Inicialización del GBM. Puede reentrenar otro modelo.
3. **Random_state**
4. **Verbose**.- Tipo de salida a ser pintada en cada modelo.
5. **Warm_start**.- Ajuste de árboles adicionales en cada modelo.
6. **Presort** .- Preselección de datos para divisiones más rápidas y certeras.



EXTREME GRADIENT BOOSTING

- Tiempo después de la publicación del algoritmo de Gradient Boosting, se le incorporó muchas propiedades avanzadas y de optimización.
- Como resultado de éstas propiedades nació XGBOOST, uno de los algoritmos más usados en competiciones, en la industria o en problemas específicos de rápida solución.



VENTAJAS DEL XGBOOST VS GBM

➤ REGULARIZACIÓN:

- ✓ La implementación estándar de GBM no tiene regularización como XGBoost, por lo tanto, también ayuda a reducir el sobreajuste.
- ✓ De hecho, XGBoost también se conoce como técnica de "refuerzo **regularizado**".

➤ PROCESAMIENTO EN PARALELO:

- XGBoost implementa el procesamiento paralelo y es **sorprendentemente más rápido** en comparación con GBM.
- XGBoost también admite la implementación en Hadoop.

➤ ALTA FLEXIBILIDAD

- XGBoost permite a los usuarios definir **objetivos de optimización personalizados y criterios de evaluación**.
- Esto agrega una nueva dimensión al modelo y no hay límite para lo que podemos hacer.

VENTAJAS DEL XGBOOST VS GBM

➤ MANEJO DE VALORES PERDIDOS

- ✓ XGBoost tiene una rutina incorporada para manejar los valores perdidos.

➤ PODA DE ARBOLES:

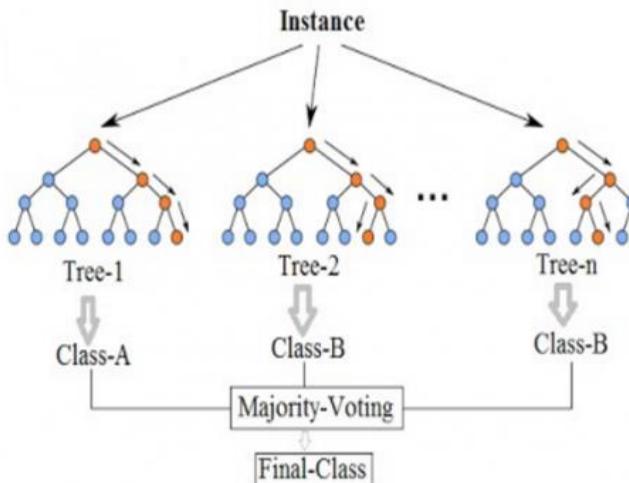
- ✓ Un GBM dejaría de dividir un nodo cuando encuentre una pérdida negativa en la división. Por lo tanto, es más un **algoritmo codicioso**.
- ✓ XGBoost por otro lado hace **divisiones hasta la max_depth** especificada y luego comienza a **podar** el árbol hacia atrás y eliminar divisiones más allá de las cuales no hay ganancia positiva.

➤ CROSS-VALIDATION INCORPORADO

- ✓ XGBoost permite al usuario ejecutar una **validación cruzada en cada iteración** del proceso de refuerzo y, por lo tanto, es fácil obtener el número óptimo exacto de iteraciones de refuerzo en una sola ejecución.

PARÁMETROS DE XGBOOST (EXTREME GRADIENT BOOSTING)

- **PARÁMETROS GENERALES:** Afectan a cada árbol individual en el modelo.
- **PARÁMETROS DE REFUERZO O BOOSTING:** Afectan la operación de **refuerzo** en el modelo.
- **PARÁMETROS DE APRENDIZAJE DE TAREAS :** Otros parámetros para el funcionamiento general.



PARÁMETROS DE XGBOOST : PARÁMETROS GENERALES

1. **Booster** .- Tipo de modelo a elegir. Árbol o Lineal.
2. **Silent** .- Mensajes o avisos mientras el modelo se ajusta.
3. **Nthread** .-Número de núcleos a usar del sistema.

PARÁMETROS DE XGBOOST : PARÁMETROS DE BOOSTING

4. **Eta** .- Ratio de aprendizaje o contribución de cada árbol.
5. **Min_child_weight**.- Suma mínima de pesos de las observaciones.
6. Similar a `min_child_leaf` en GBM.
7. **Max_depth** .-
8. **Max_leaf_nodes**
9. **Gamma**.- Mínima reducción del error requerida para una división o corte de nodo.
10. **Max_delta_step**.- Actualización de pesos. Estilo conservador.
11. **Subsample**.- Fracción de observaciones en cada árbol.
12. **Colsample_bytree**.- Equivalente a `max_features`.
13. **Colsample_bylevel**.- Proporción de variables para cada corte.
14. **Lambda**.- Regularización L2. (Ridge)
15. **Alpha**.- Regularización L1. (Lasso)
16. **Scale_pos_weight**.- En caso de desequilibrio ayuda a la convergencia.

PARÁMETROS DE XGBOOST : PARÁMETROS APRENDIZAJE DE TAREAS

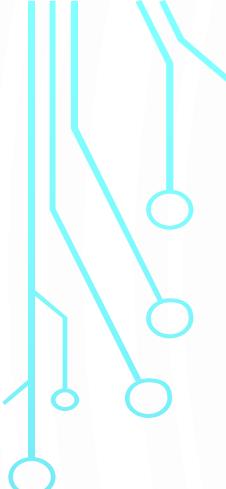
1. Objective .-

- ✓ **reg:linear**
- ✓ **binary:logistic**
- ✓ **multi:softmax**
- ✓ **multi:softprob**

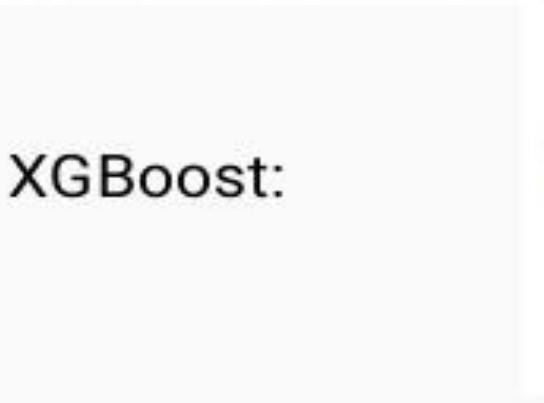
2. Eval_metric .- Métrica usada en la validación de la data.

- ✓ **Rmse** – root mean square error
- ✓ **Mae** – mean absolute error
- ✓ **Logloss** – negative log-likelihood
- ✓ **Error** – Binary classification error rate (0.5 threshold)
- ✓ **Merror** – Multiclass classification error rate
- ✓ **Mlogloss** – Multiclass logloss
- ✓ **Auc**: Area under the curve

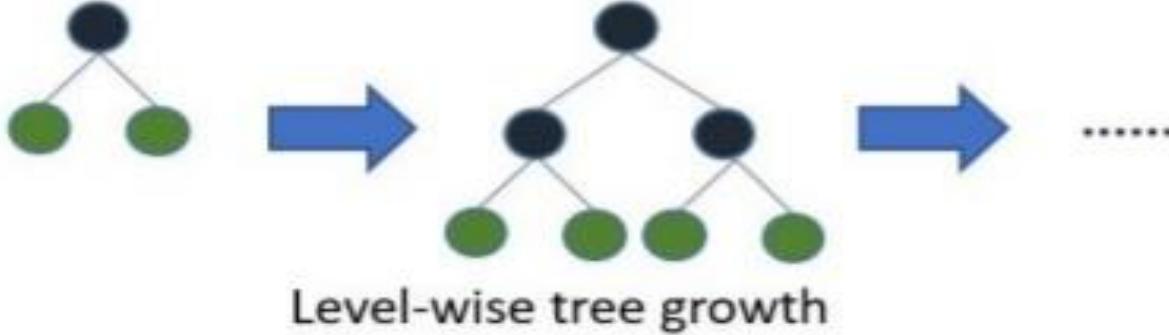
3. Seed .-Semilla aleatoria.



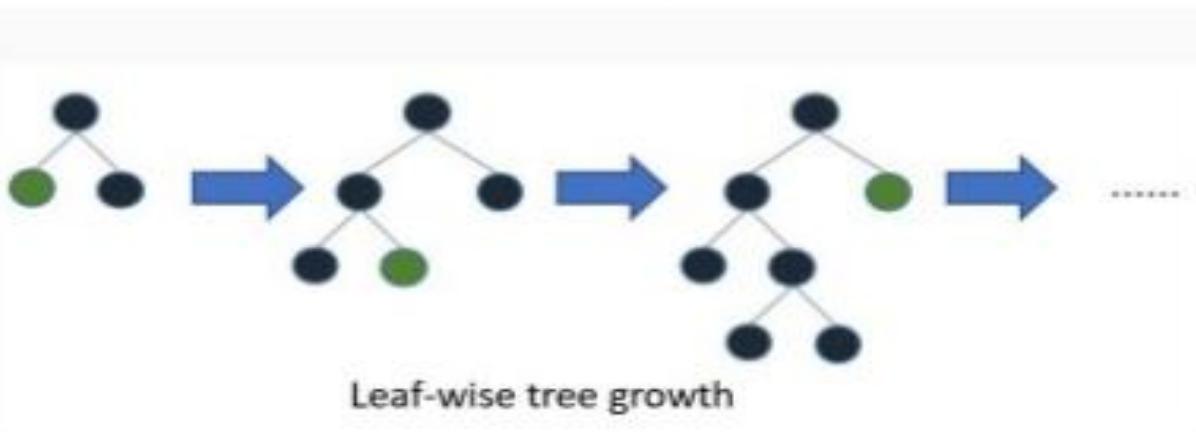
LIGHTGBM



XGBoost:



LightGBM:



QUÉ ES LIGHTGBM ?

- Es un **Gradient Boosting Algorithm** basado en árboles de decisión, rápido, distribuido y de alto rendimiento
- Búsqueda los mejores ajustes a nivel de hojas, es decir su **búsqueda es por hoja y no por nivel.**
- Debido a que su búsqueda es por hoja, puede mejorar y reducir más pérdidas que el algoritmo de nivel , por ende en muchos casos llega a ser más preciso.
- Debido a su búsqueda por hojas llega a una rapidez sorprendente :
Light = Luz.

VENTAJAS DEL LIGHTGBM

- **Mayor velocidad de entrenamiento y mayor eficiencia.**
- **Menor uso de memoria:** Reemplaza valores continuos a bins discretos que resultan en un menor uso de memoria.
- **Mejor precisión que cualquier otro algoritmo de impulso.**
- **Compatibilidad con grandes conjuntos de datos:** Es capaz de funcionar igual de bien con grandes conjuntos de datos con una reducción significativa en el tiempo de entrenamiento en comparación con XGBOOST.
- **Aprendizaje en paralelo compatible.**

PARÁMETROS DE LIGHTGBM

- **Application:** default=regression.
 - Regression : perform regression task
 - Binary : Binary classification
 - Multiclass: Multiclass Classification
- **Num_iterations:** Número de árboles en el boosting.
- **Num_leaves :** Número de hojas por árbol.
- **Device:** Dispositivo CPU o GPU.
- **Max_depth:**
- **Min_data_in_leaf:**
- **Feature_fraction:** Proporción de los features por árbol.
- **bagging_fraction:** Proporción de la data por árbol.
- **min_gain_to_split.**
- **Max_bin :** Max number of bins to bucket the feature values.
- **min_data_in_bin :** Min number of data in one bin
- **Label.**
- **Categorical_feature.**

CATBOOST

- **CatBoost** es un algoritmo de aprendizaje de máquina de fuente abierta recientemente de **Yandex**.
- Se puede integrar fácilmente con los marcos de aprendizaje profundo como **Google TensorFlow** y **Apple's Core ML**.
- Puede trabajar con diversos tipos de datos para ayudar a resolver una amplia gama de problemas que las empresas enfrentan en la actualidad.
- **CatBoost** = **Cat** (Categorical) + **Boost** (Boosting)

<https://tech.yandex.com/catboost/>

XGBOOST - LIGHTGBM - CATBOOST

Function	XGBoost	CatBoost	Light GBM
Important parameters which control overfitting	<ul style="list-style-type: none"> 1. learning_rate or eta – optimal values lie between 0.01-0.2 2. max_depth 3. min_child_weight: similar to min_child_leaf; default is 1 	<ul style="list-style-type: none"> 1. Learning_rate 2. Depth - value can be any integer up to 16. Recommended - [1 to 10] 3. No such feature like min_child_weight 4. L2-leaf-reg: L2 regularization coefficient. Used for leaf value calculation (any positive integer allowed) 	<ul style="list-style-type: none"> 1. learning_rate 2. max_depth: default is 20. Important to note that tree still grows leaf-wise. Hence it is important to tune num_leaves (number of leaves in a tree) which should be smaller than $2^{(max_depth)}$. It is a very important parameter for LGBM 3. min_data_in_leaf: default=20, alias= min_data, min_child_samples
Parameters for categorical values	Not Available	<ul style="list-style-type: none"> 1. cat_features: It denotes the index of categorical features 2. one_hot_max_size: Use one-hot encoding for all features with number of different values less than or equal to the given parameter value (max – 255) 	<ul style="list-style-type: none"> 1. categorical_feature: specify the categorical features we want to use for training our model
Parameters for controlling speed	<ul style="list-style-type: none"> 1. colsample_bytree: subsample ratio of columns 2. subsample: subsample ratio of the training instance 3. n_estimators: maximum number of decision trees; high value can lead to overfitting 	<ul style="list-style-type: none"> 1. rsm: Random subspace method. The percentage of features to use at each split selection 2. No such parameter to subset data 3. iterations: maximum number of trees that can be built; high value can lead to overfitting 	<ul style="list-style-type: none"> 1. feature_fraction: fraction of features to be taken for each iteration 2. bagging_fraction: data to be used for each iteration and is generally used to speed up the training and avoid overfitting 3. num_iterations: number of boosting iterations to be performed; default=100

GRACIAS POR SU ATENCIÓN

