

DATA SCIENCE

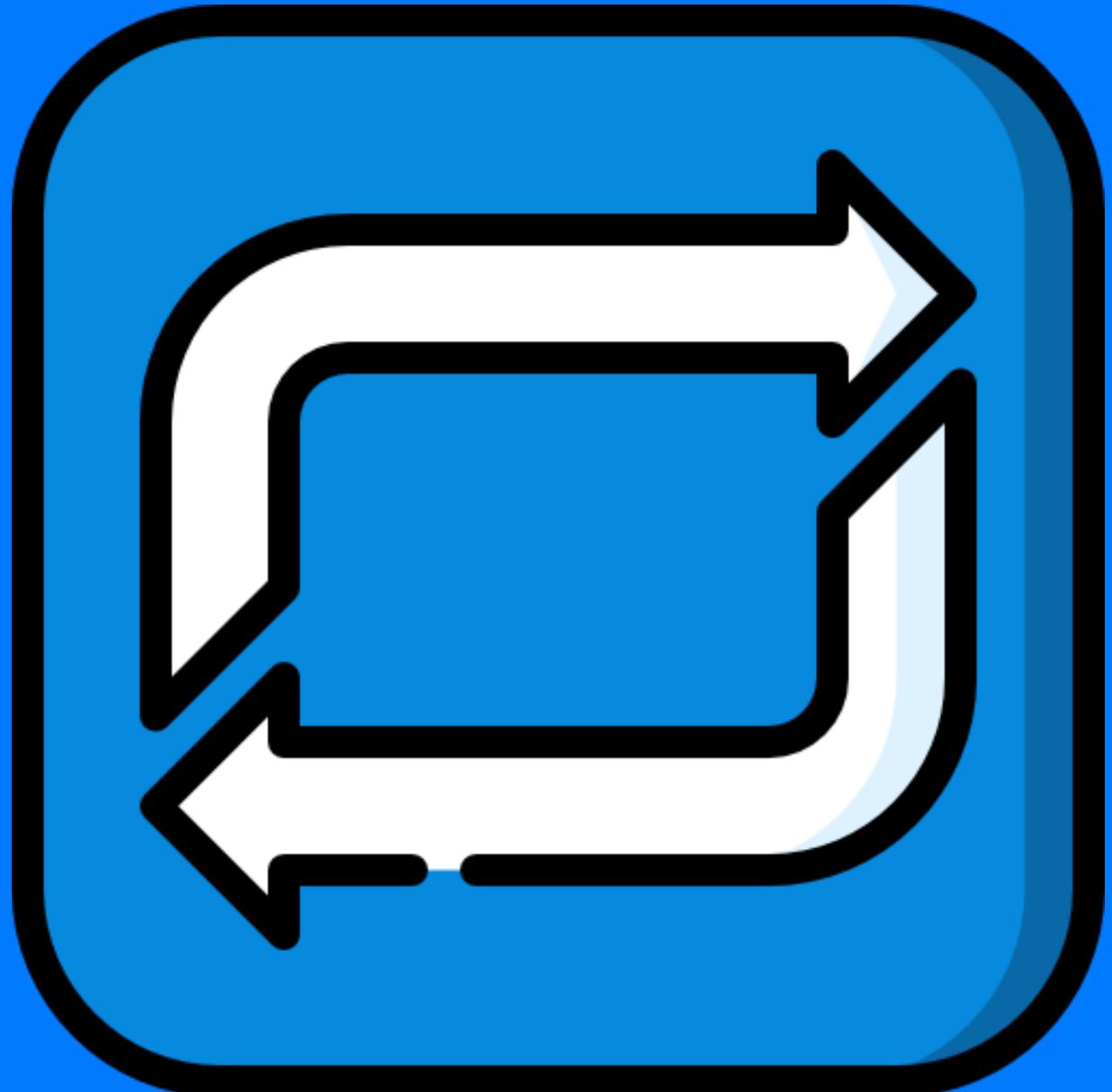


Federico Baiocco
baioccofede@gmail.com
3512075440



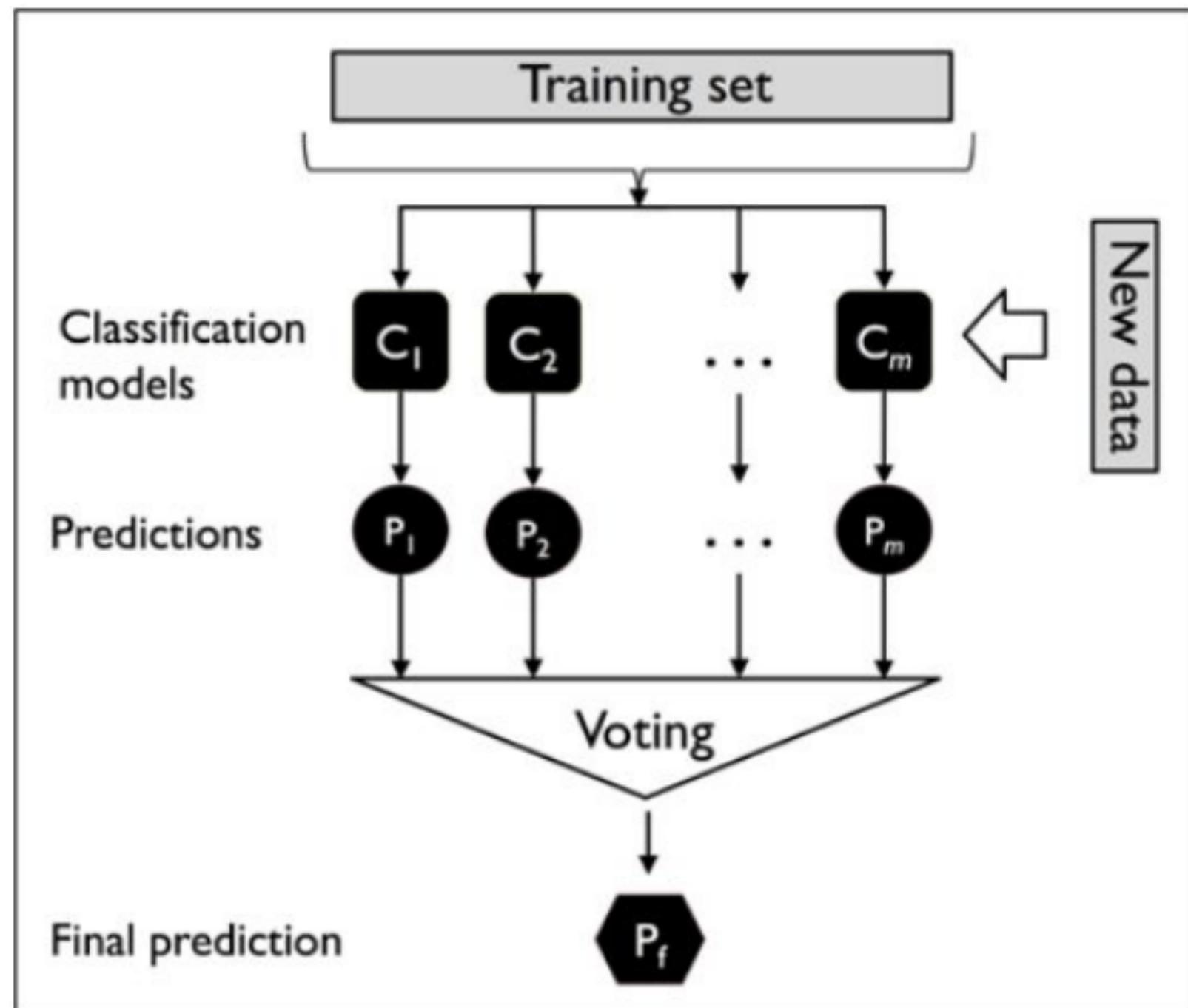
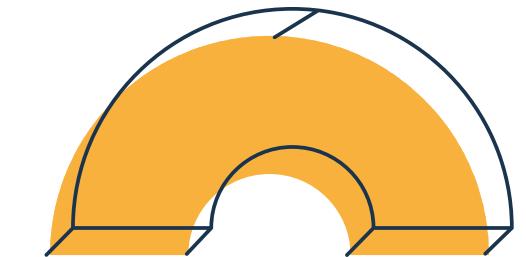
Clase 26 - Agenda

ENSAMBLES - BOOSTING

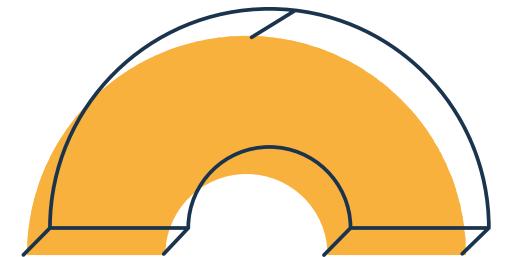


Repaso

Ensambles

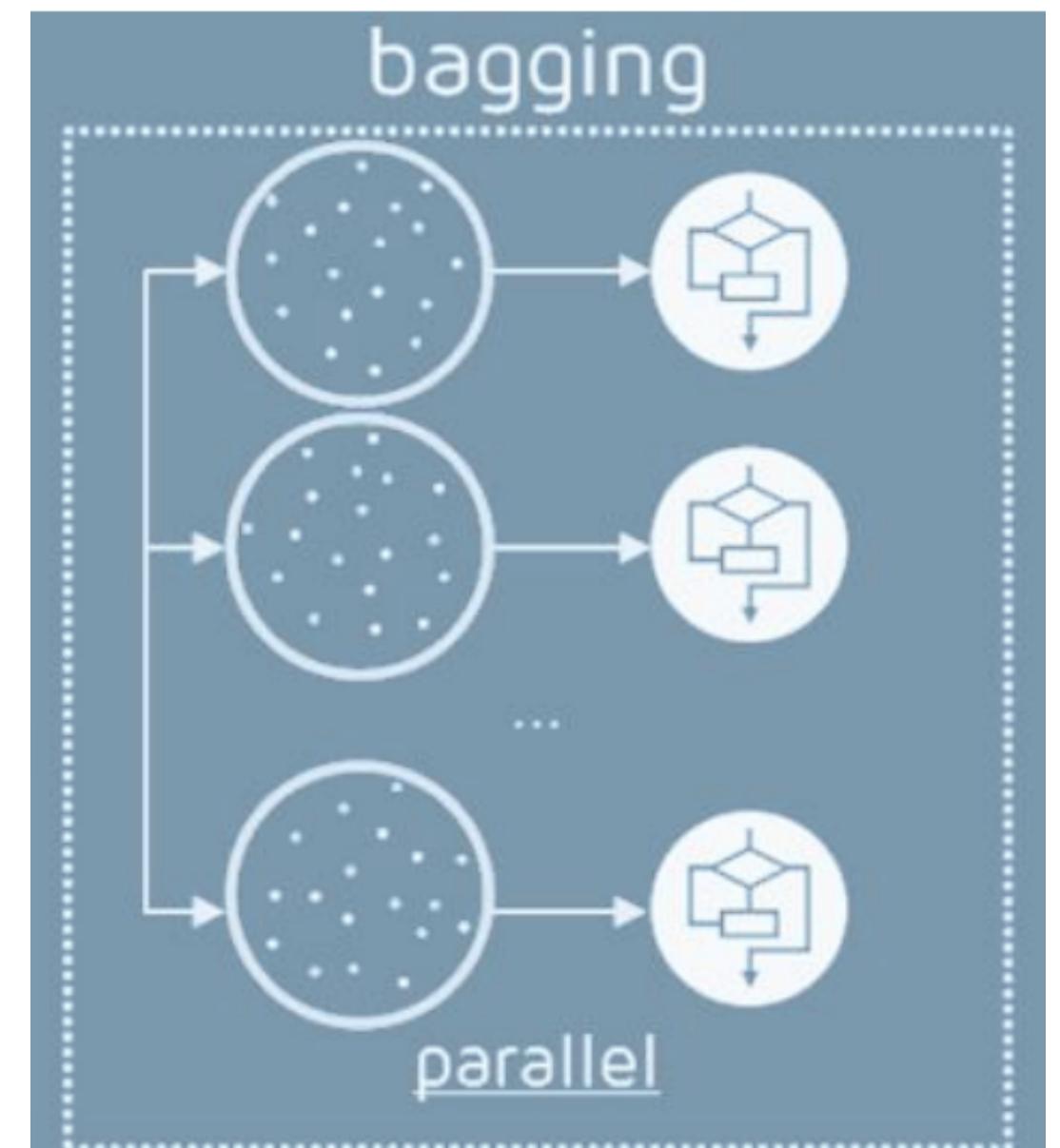


Bagging

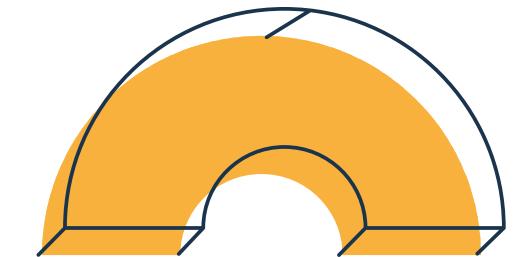


Bagging (o Bootstrap Aggregation) :

- Dada una muestra de datos, se extraen varias muestras
- Esta selección se realiza de manera aleatoria
- Una vez que forman las muestras, se entranan modelos de manera separada.
- La predicción final se hace por "votación"

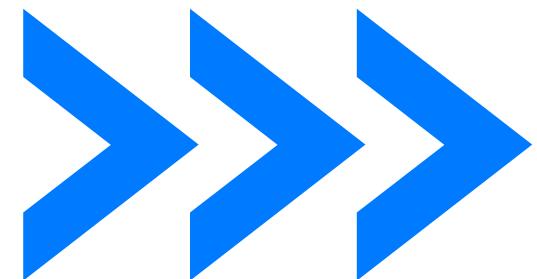


Random forest



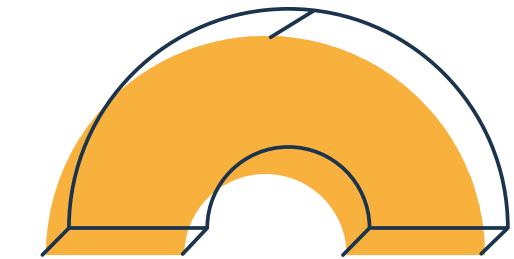
1) Creamos un dataset a partir de bootstrapping: Se seleccionan filas de manera aleatoria. Puede seleccionarse la misma fila más de una vez.

Gender	Age	EstimatedSalary	Purchased
Male	26	32000	0
Male	37	72000	0
Female	49	36000	1
Male	25	33000	0

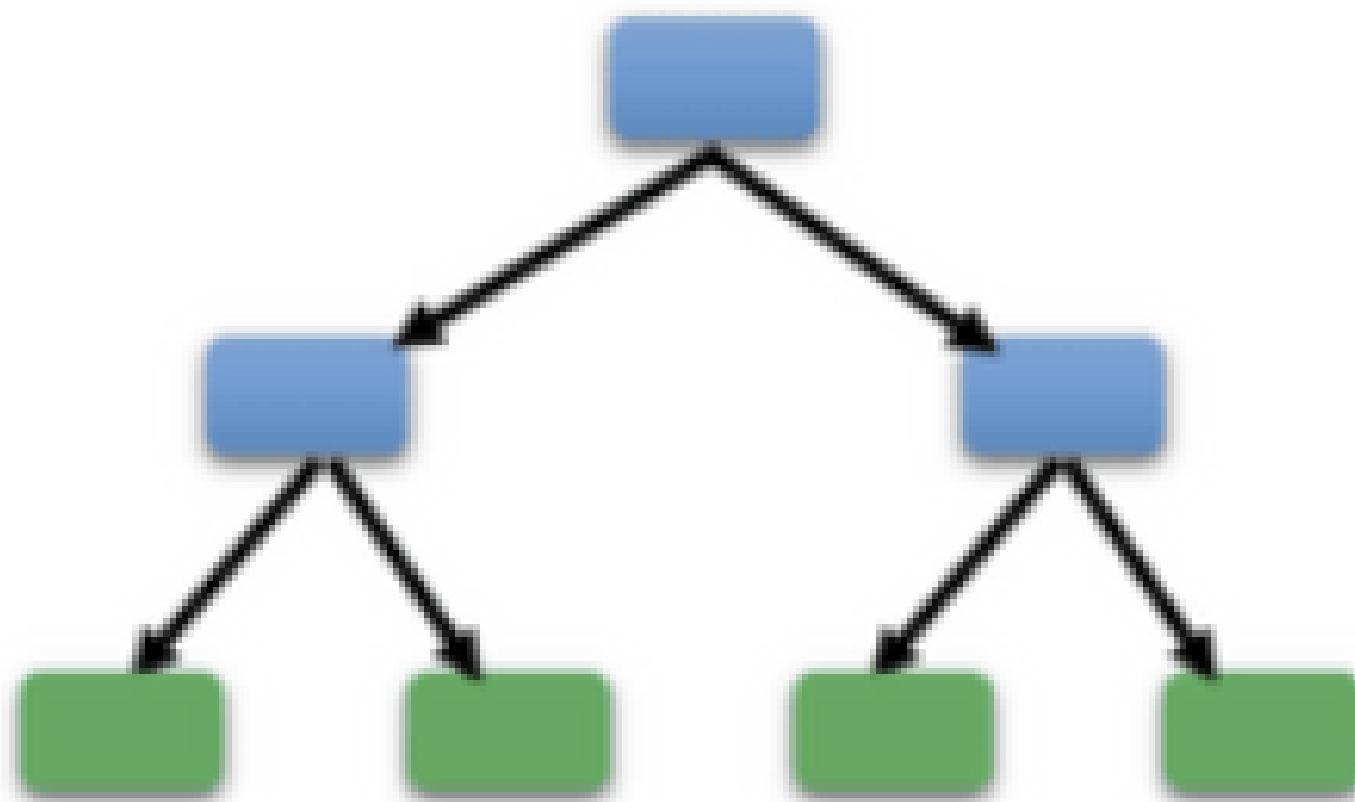


Gender	Age	EstimatedSalary	Purchased
Male	37	72000	0
Male	26	32000	0
Male	25	33000	0
Male	25	33000	0

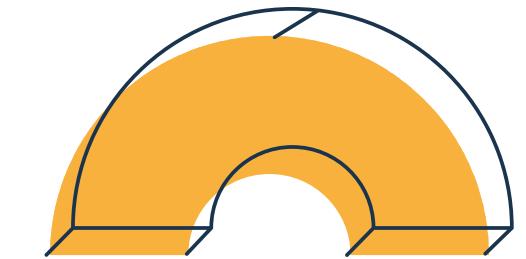
Random forest



2) Crear un decision tree utilizando el dataset creado mediante bootstrapping, pero utilizando únicamente un subset de features



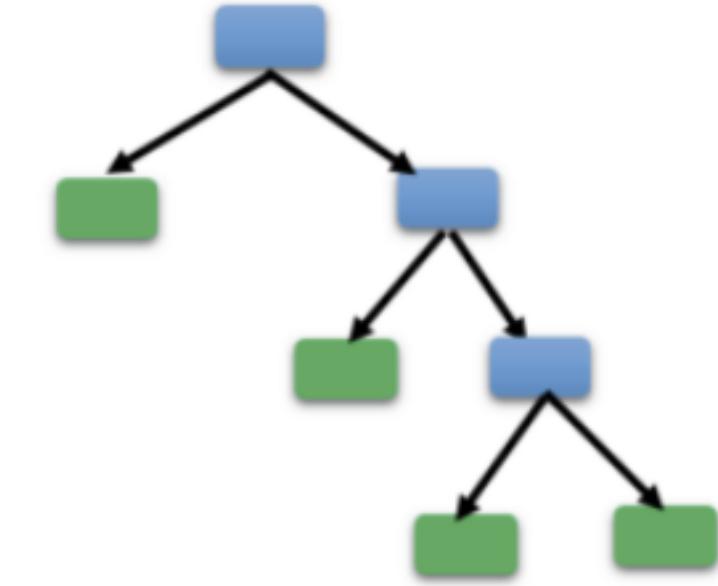
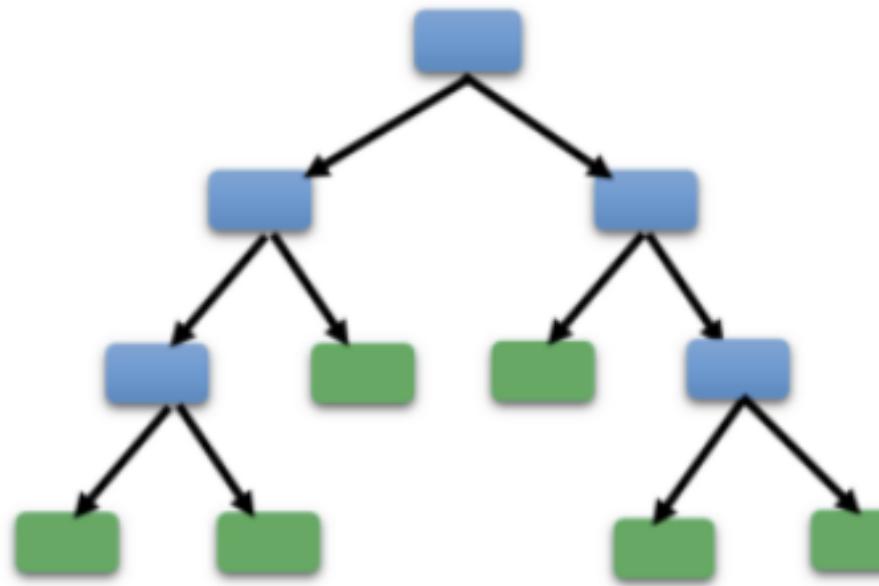
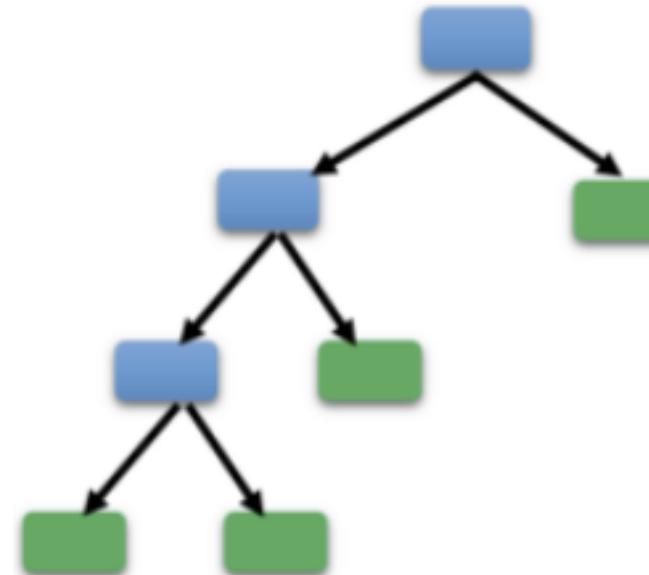
Random forest



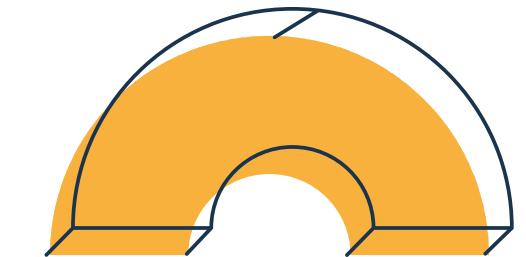
3) Vamos de nuevo al paso 1, creamos un nuevo dataset y luego un nuevo árbol (paso 2).

La idea es que entrenemos muchos (por ej: cientos) de árboles distintos.

¿ En qué van a ser distintos cada uno de los árboles?

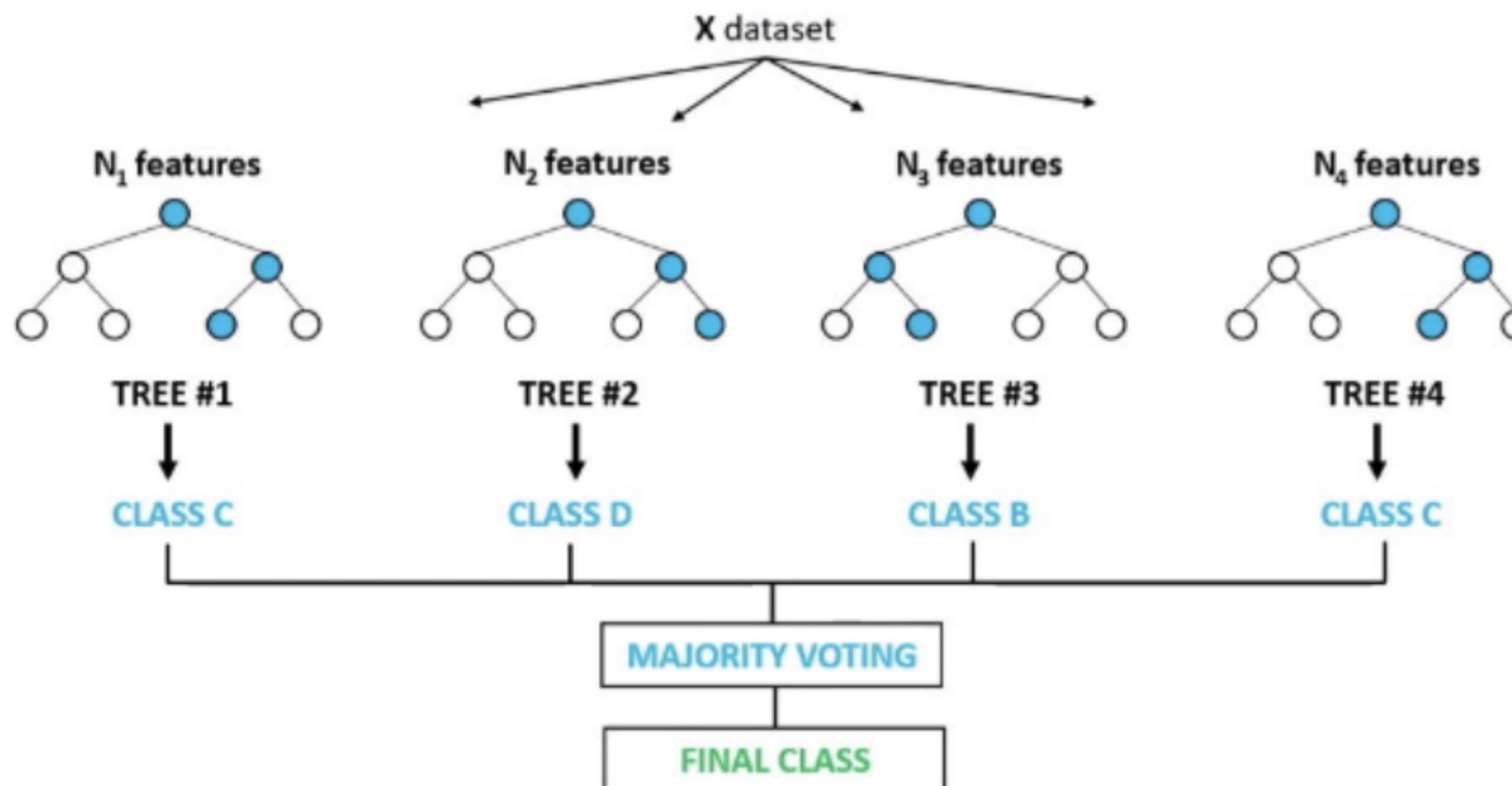


Random forest

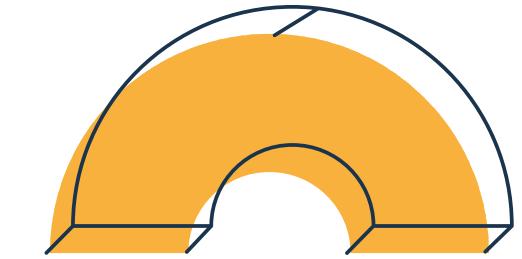


Ahora, para generar predicciones sobre nuevos datos, se utilizan todos los árboles.

Al final de todo, se hace una votación.



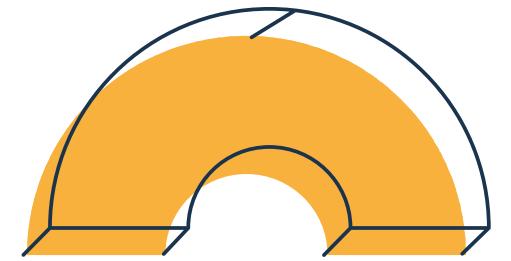
Random forest



Ventajas:

- Es un modelo robusto. (outliers)
- Luego de entrenar un random forest podemos evaluar la importancia de las distintas features
- Al entrenarse muchos árboles por separado, se puede paralelizar el proceso para que sea más rápido.
- Interpretabilidad (al estar formados por árboles de decisión)

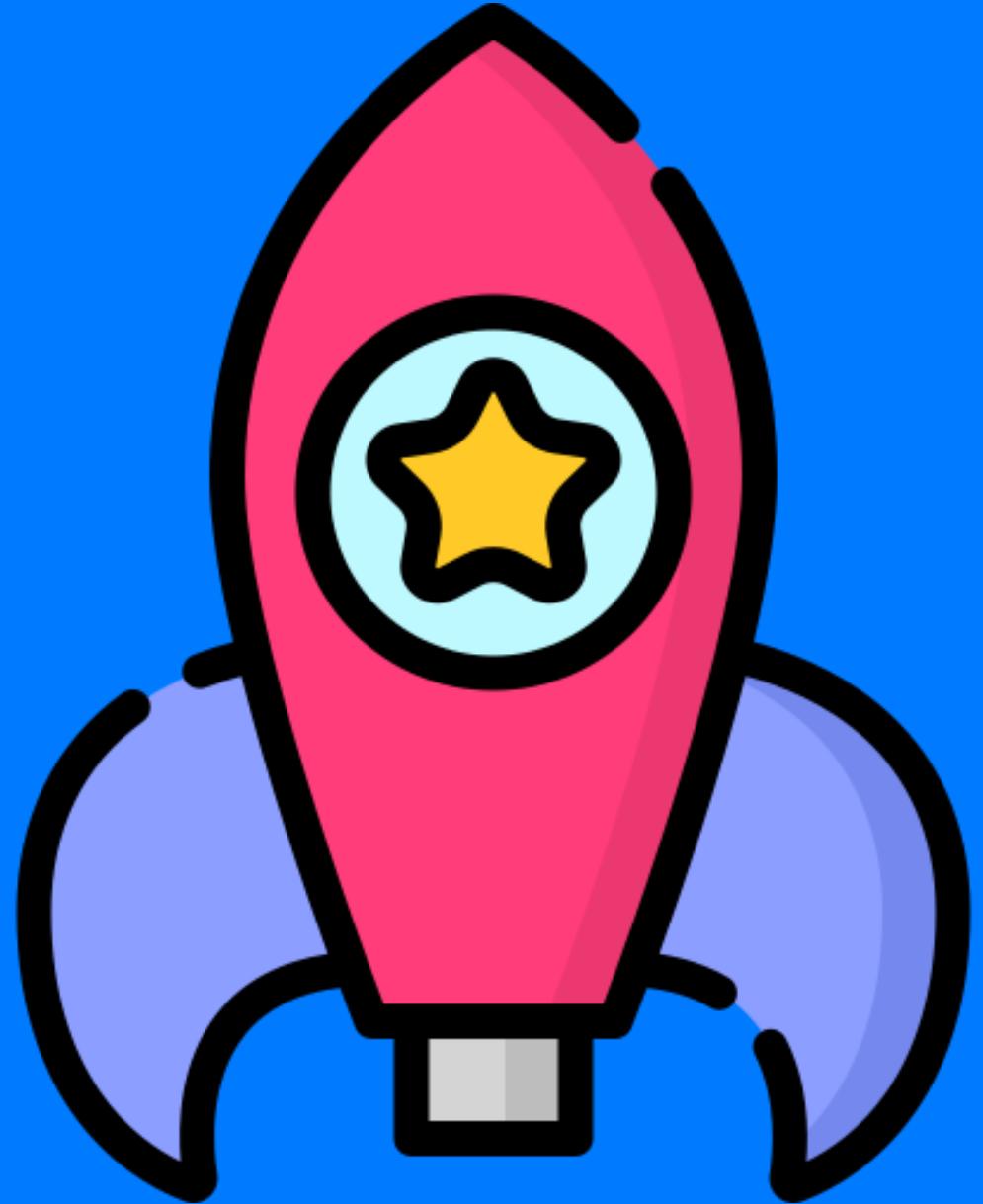
Random forest



¿ Dónde está la parte "random" de random forest ?

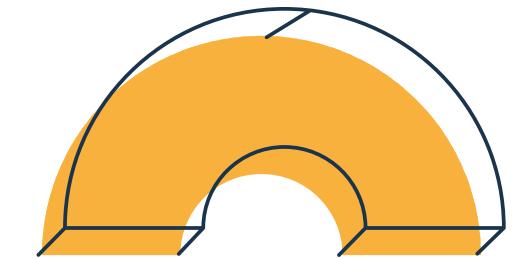
¿ Cuáles son los hiperparámetros más importantes ?

¿ Cuántas features se usan para entrenar cada árbol ?



Boosting

Boosting

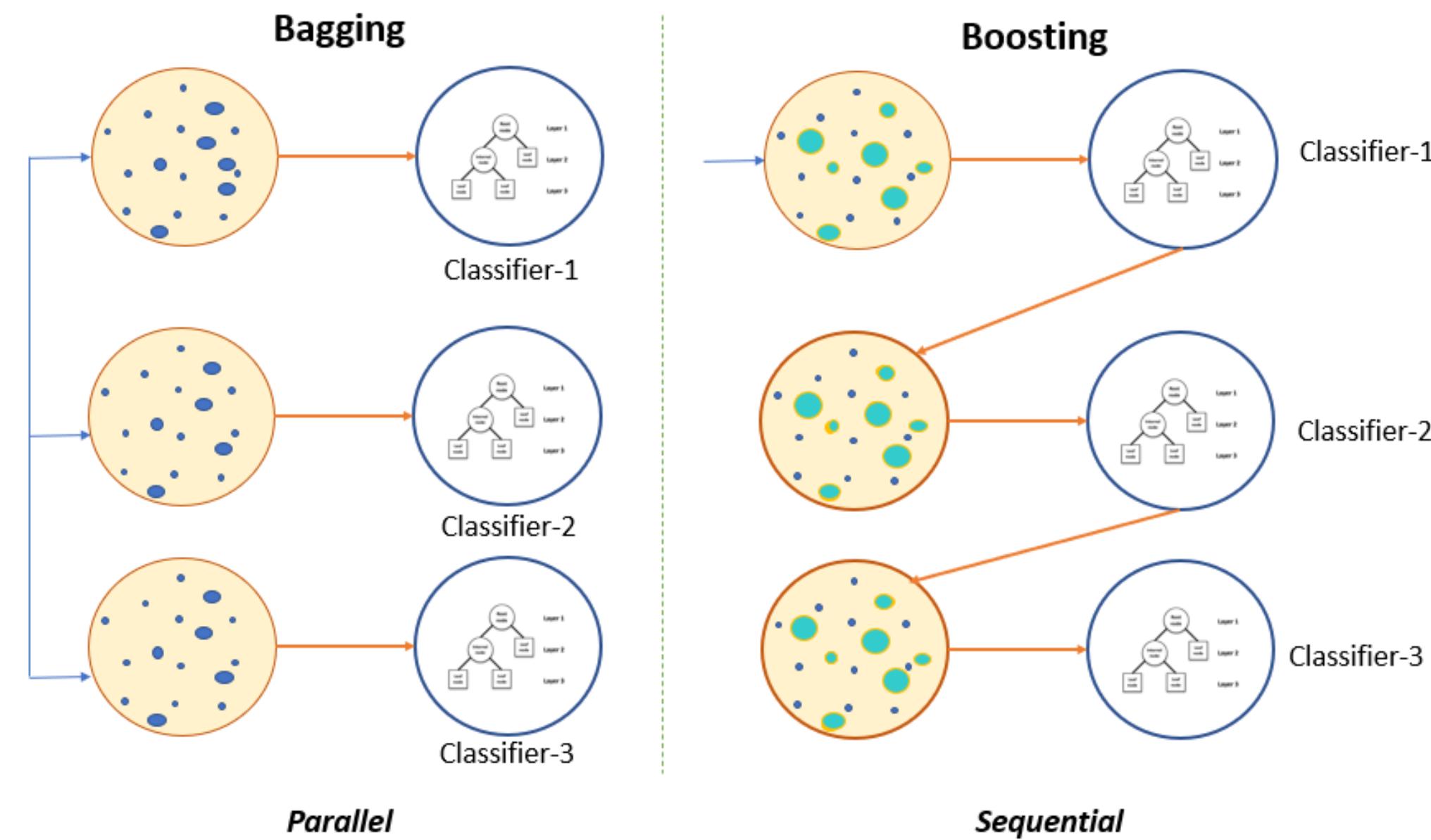
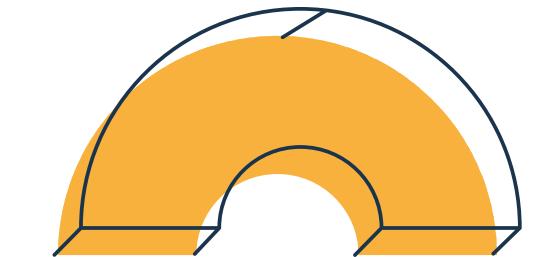


Boosting es otro método para hacer ensambles de modelos.

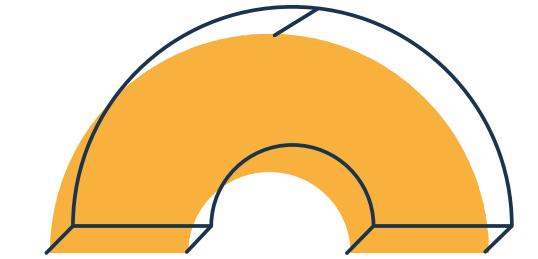
A diferencia de bagging, en este caso los modelos se ensamblan de manera secuencial

La idea principal es entrenar una secuencia de modelos en donde se le da más peso a los ejemplos que fueron clasificados erroneamente por el modelo anterior.

Boosting



Adaboost



Adaboost es una de las técnicas más populares de Boosting.

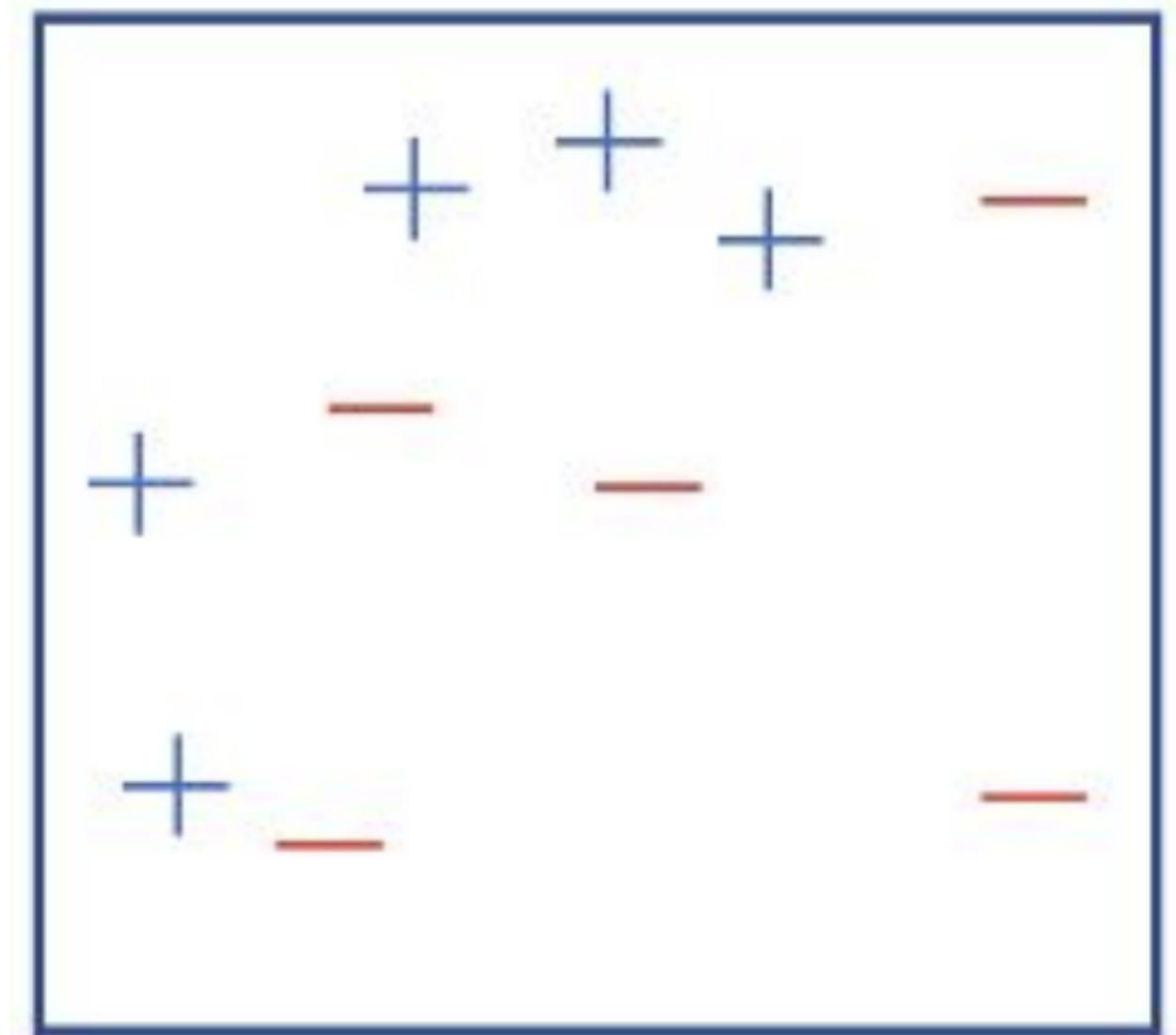
Este algoritmo, mediante un entrenamiento iterativo de clasificadores débiles, le va dando mayor importancia a los datos mal clasificados anteriormente.

Adaboost -> Adaptative boosting

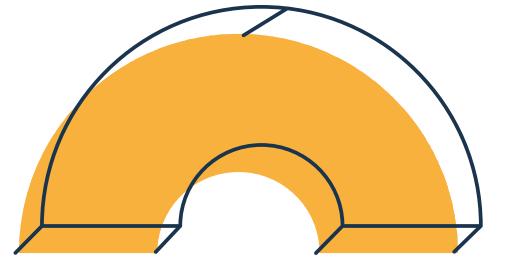
Adaboost



Para explicar un poco el funcionamiento de AdaBoost consideraremos el siguiente problema de clasificación binaria con 10 ejemplos de entrenamiento, 5 positivos y 5 negativos.



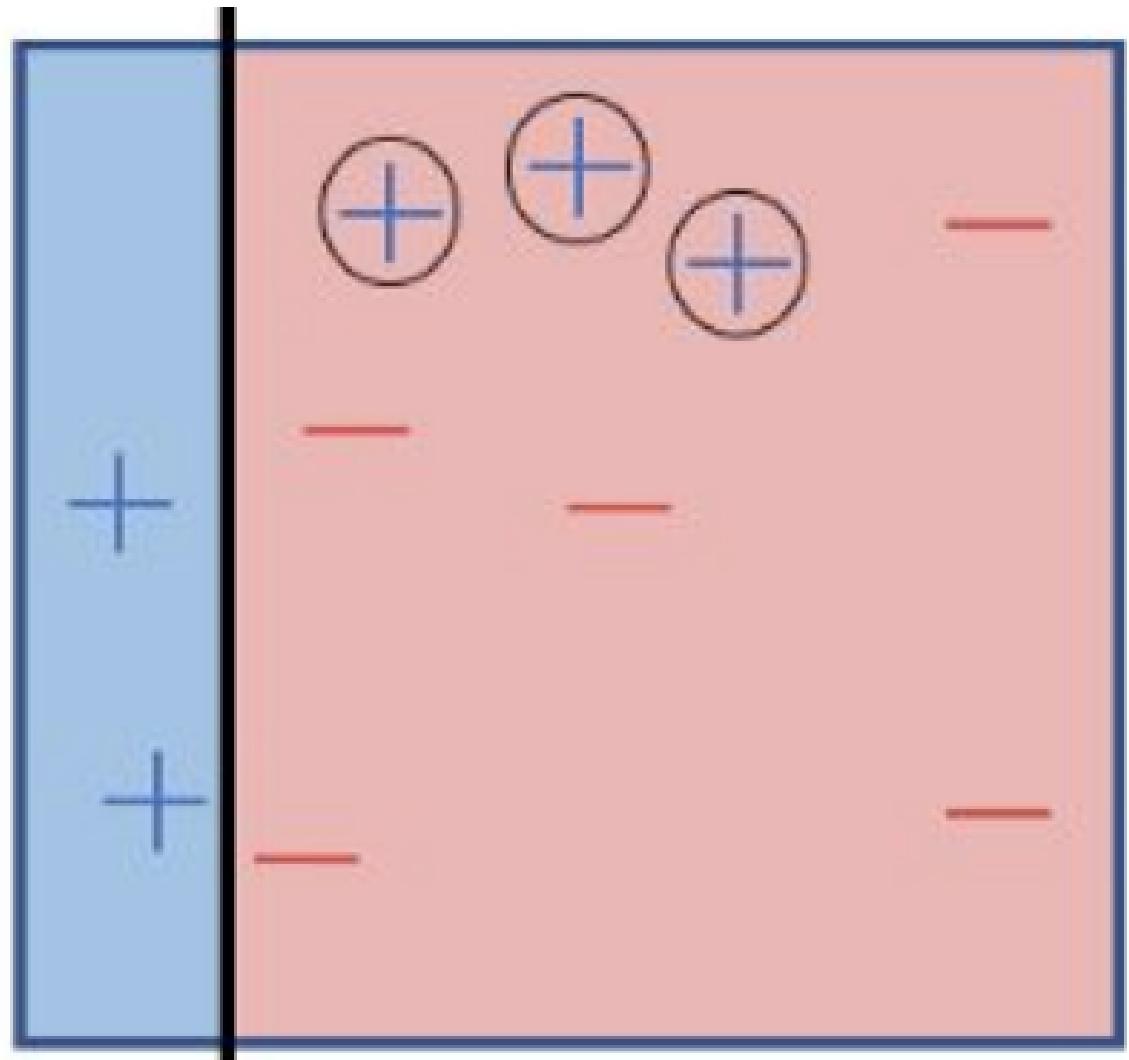
Adaboost



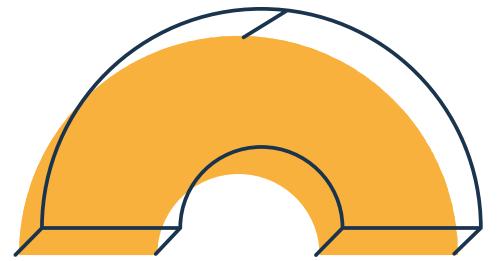
El primer clasificador débil, es una recta vertical (Un árbol de decisión con un solo nodo).

A la derecha de la recta, consideramos que todos los ejemplos son negativos, mientras que a la izquierda son positivos.

La recta clasifica mal a tres positivos.

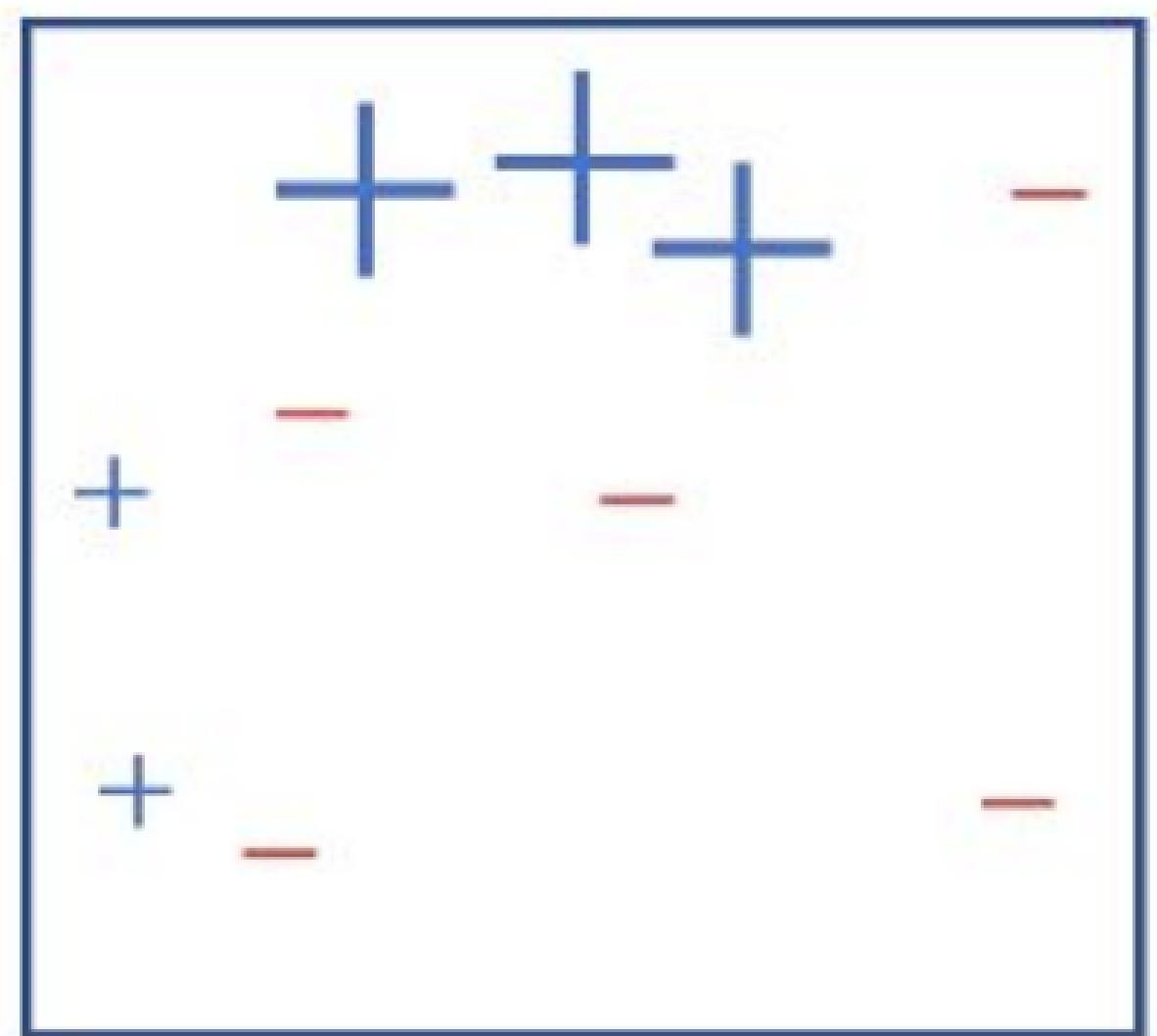


Adaboost

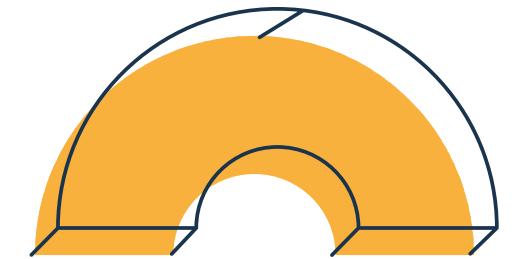


Acá vemos como los tres ejemplos mal clasificados aparecen con un tamaño mayor que el resto de los ejemplos.

Esto significa que dichos ejemplos tendrán una mayor importancia al momento de seleccionar el clasificador débil de la segunda iteración.

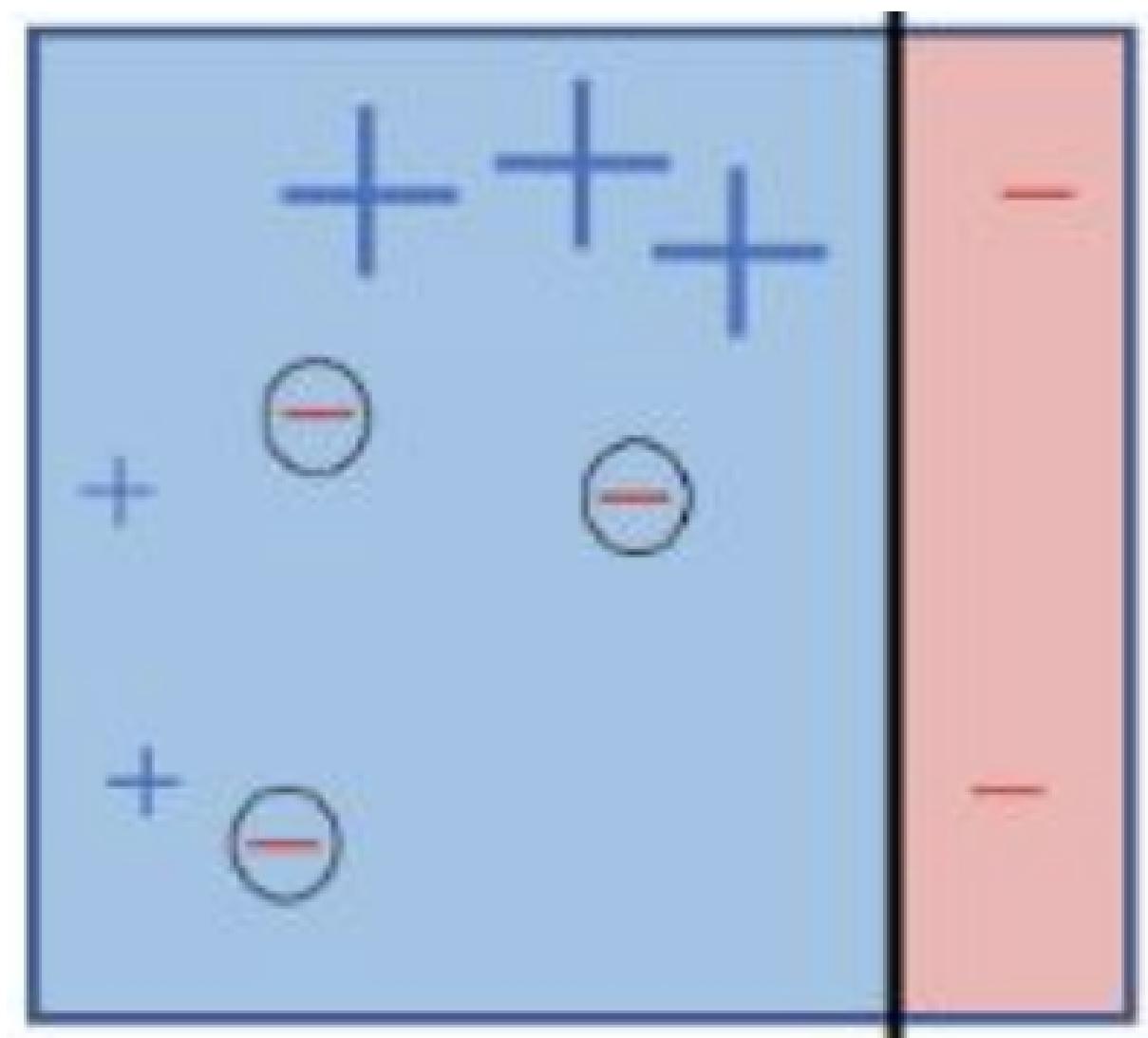


Adaboost

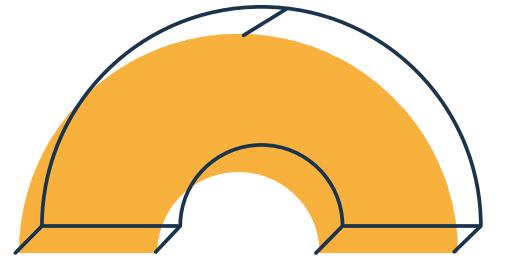


El clasificador débil, es otra recta vertical colocada más hacia la derecha.

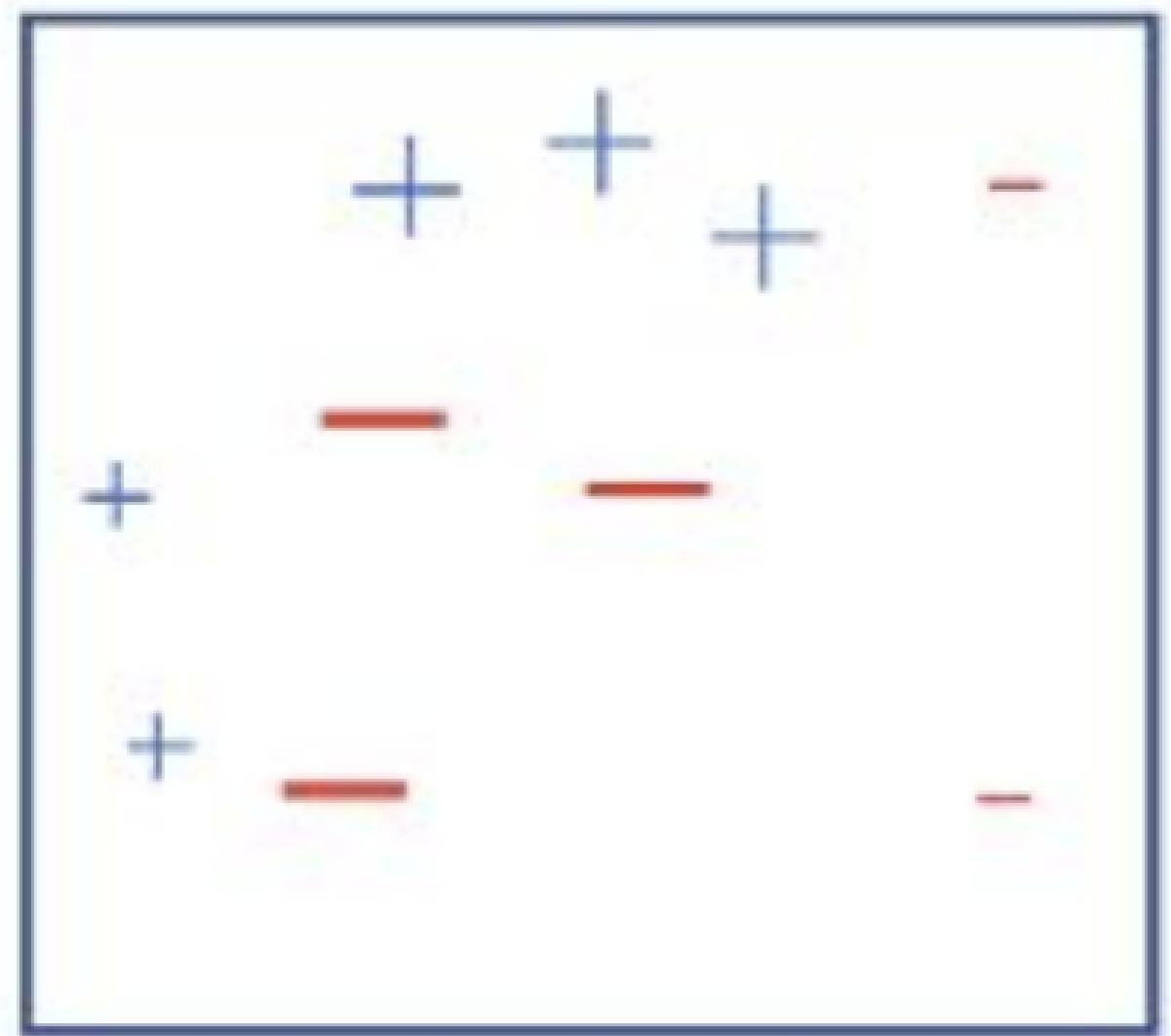
Este segundo clasificador se equivoca también en tres ejemplos (negativos).



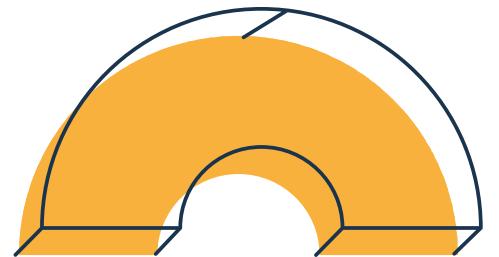
Adaboost



Acá vemos que para la tercera iteración los ejemplos negativos mal clasificados tienen ahora el mayor tamaño, es decir, tendrán mayor importancia en la siguiente iteración.

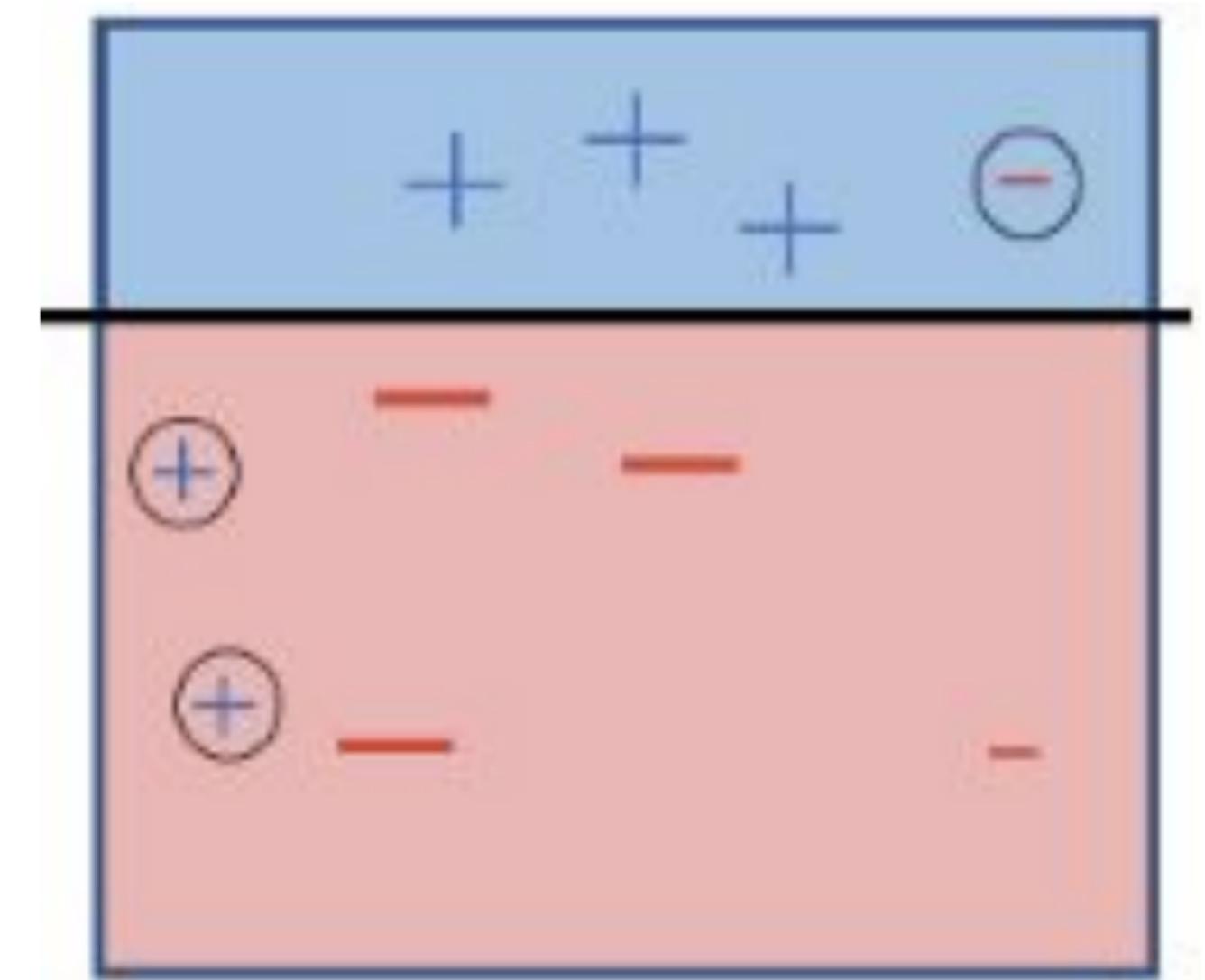


Adaboost

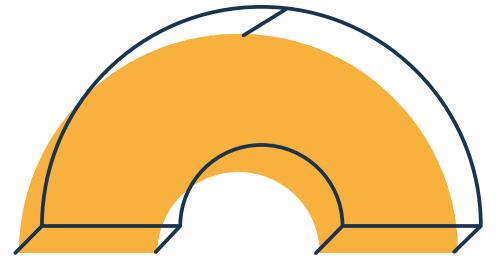


En la tercera iteración el clasificador débil resultante es una recta horizontal, como se puede observar en el cuadro de la derecha.

Este clasificador se equivoca en la clasificación de un ejemplo negativo y dos positivos, que de igual forma aparecen encerrados en un círculo.

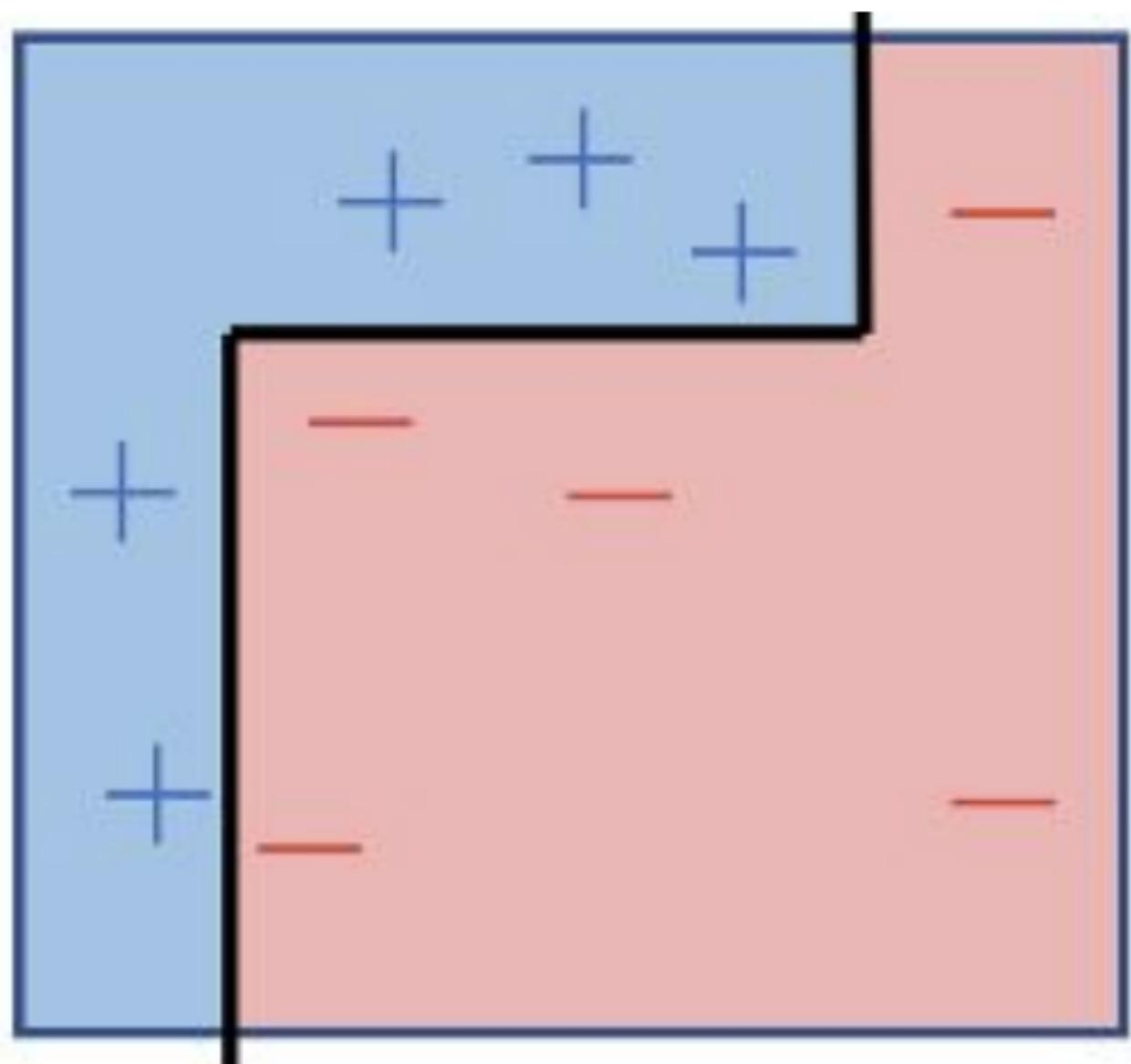


Adaboost



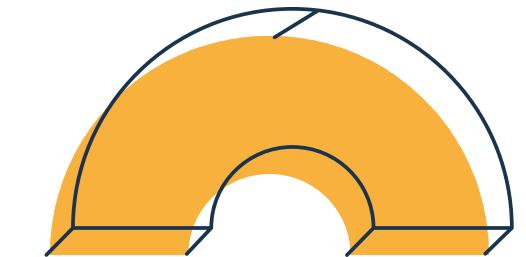
Finalmente, vemos el clasificador "fuerte" que resulta de crear un ensamble con tres clasificadores débiles. La forma en que utilizamos estos tres clasificadores débiles es mediante una decisión por mayoría.

Cuando vamos a clasificar un nuevo ejemplo, le preguntamos a cada uno de nuestros tres clasificadores débiles su opinión. Si la mayoría opina que el nuevo ejemplo es positivo, la decisión del clasificador fuerte será que es un ejemplo positivo.



XGBoost

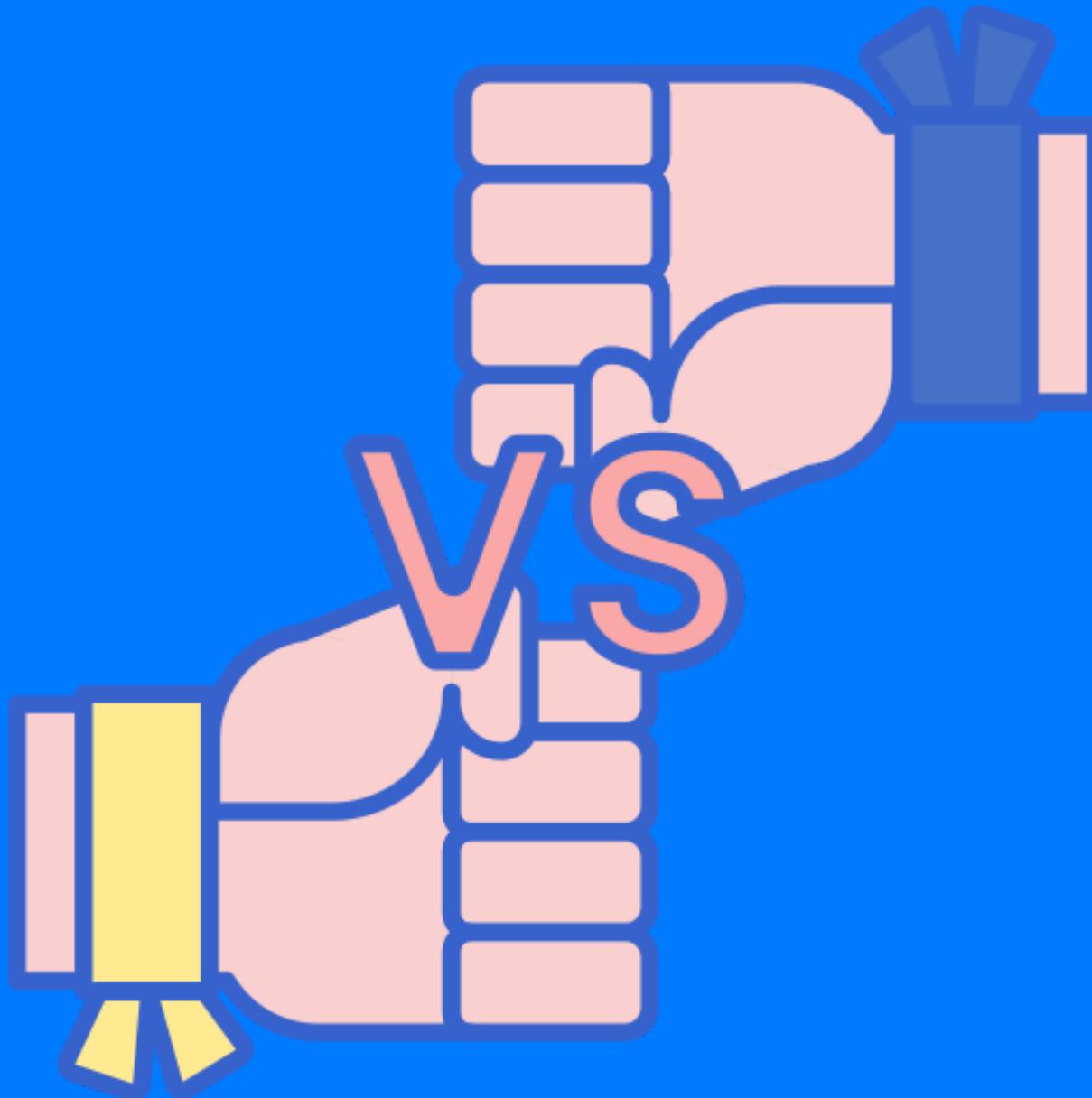
XGBOOST



XGBOOST es un modelo que actualmente domina el aprendizaje automático, famoso por ganar muchas competencias de ML.

Utiliza un enfoque de boosting, donde nuevos modelos se van creando de manera secuencial, pero además utiliza el algoritmo del "descenso por gradiente" que veremos más adelante (junto a redes neuronales).

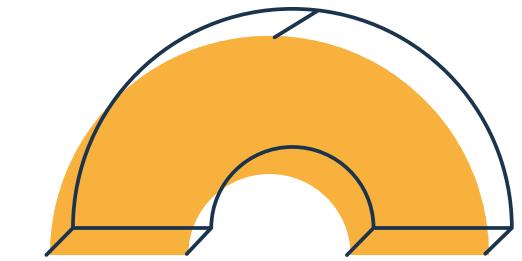
Tiene una muy buena performance (es rápido) y funciona muy bien para datos estructurados (como los que venimos utilizando hasta el momento)



Bagging vs boosting

¿Cuál es mejor?

Bagging vs Boosting

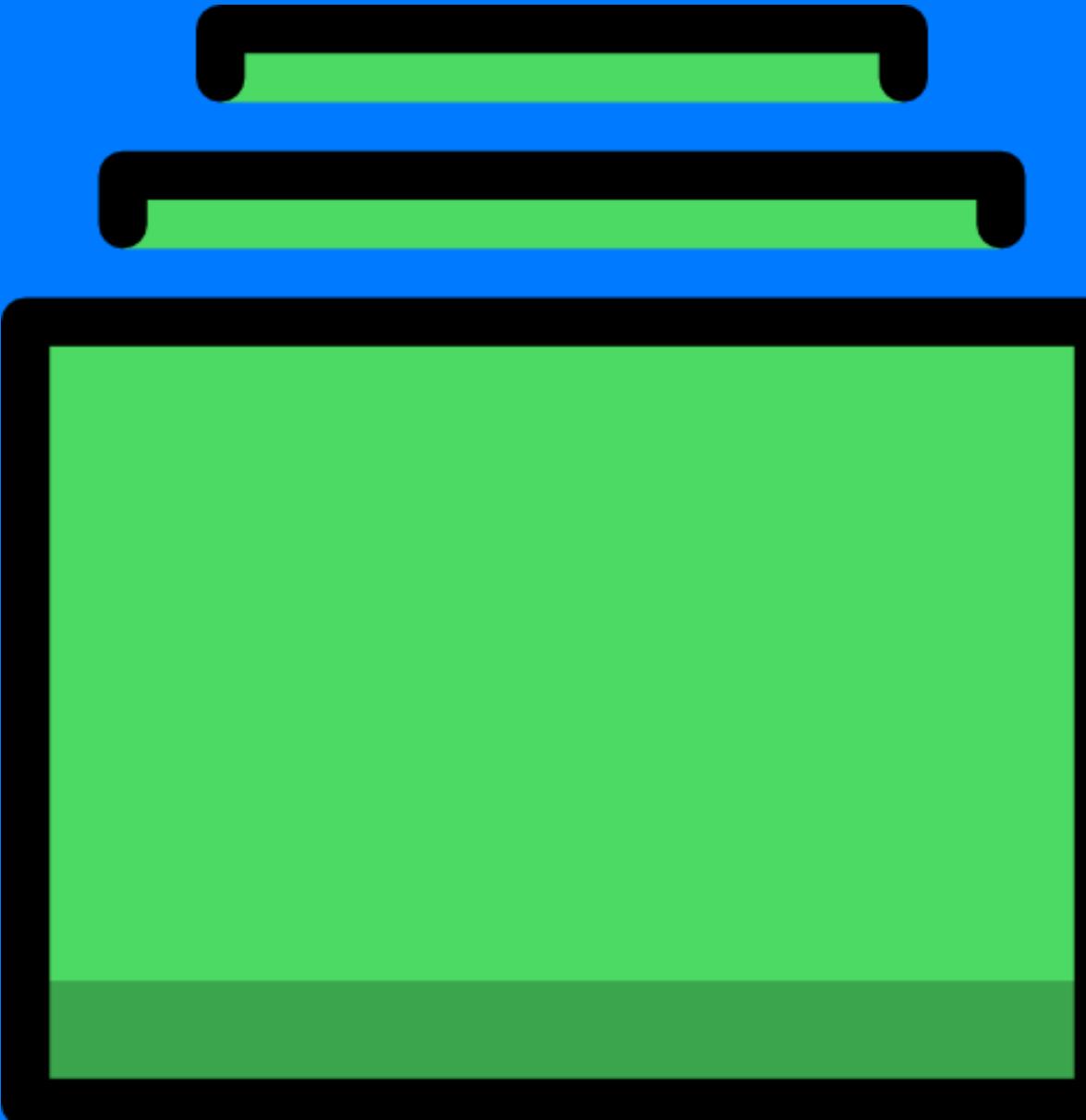


Bagging

- Se entranan modelos de manera independiente
- Fácilmente paralelizable
- Ayuda a prevenir el overfitting

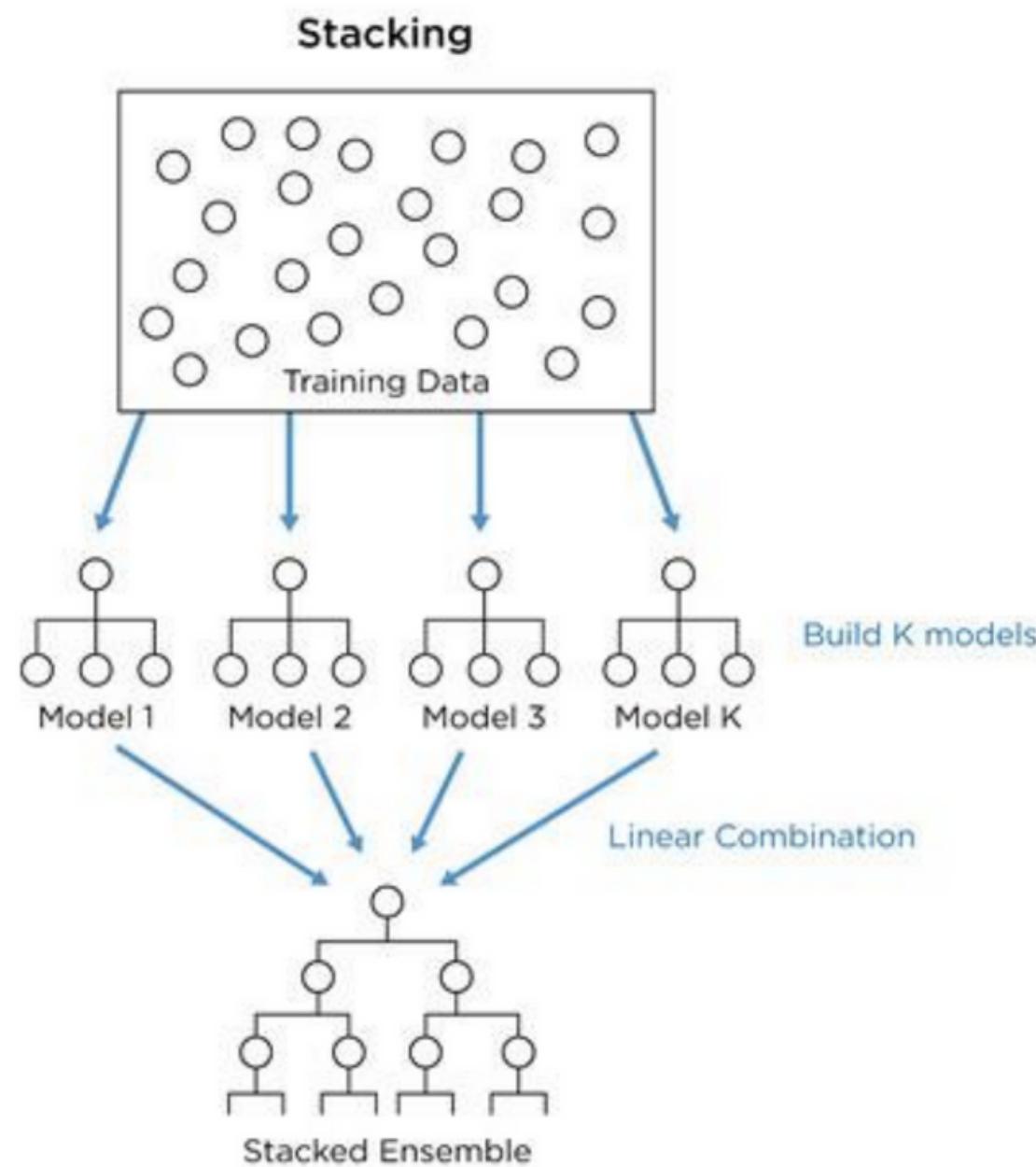
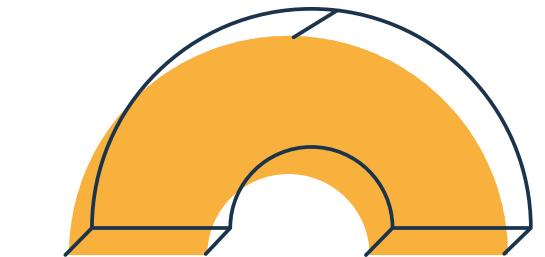
Boosting

- Modelos entrenados de manera secuencial, donde cada uno mejora las predicciones del anterior.
- Propenso a overfittear
- No se puede paralelizar fácilmente



Stacking

Stacking



Otra forma de hacer ensambles de modelos es stacking:

Cuando hacemos stacking, la idea es entrenar distintos modelos (pueden ser todos distintos). Cada uno de estos modelos genera sus propias predicciones.

Una vez que tenemos las predicciones de todos los modelos, utilizamos un "final estimator" que tomará como features (X) las predicciones de todos los modelos anteriores.



BREAK!