Ensambles de modelos

Teorica 11/10

- [ ] https://drive.google.com/file/d/1asLg1x5518lr-bevy48f7y1nAkyQpbm4/view
- [ ] https://drive.google.com/file/d/1XNXcL-RzXLHkx0t4QBqd9BRGzuquSw8k/view

Practica 13/10

- [ ] Random Forest y Boosting
  https://drive.google.com/file/d/1NUc8PT9T193Fzrb6_hLvxe3sF6oRV1ZC/view
- [ ] Voting y Stacking
  https://drive.google.com/drive/folders/1isWl3TYwyhvDTY17nw-Ded7-Z2zPY_po

- Bias (sesgo)
- Varianza
- Underfitting y overfitting
- Bagging
- AdaBoost
- Gradient Boost
- XGBoost
- Ensambles Hibridos
  - Voting
  - Stacking
  - Cascading

La sabiduría de las multitudes

En 1906 Galton (primo de Darwin) quería demostrar lo “bruta” que era la gente en base a un experimento. Este constaba de adivinar el peso de una vaca en una exposición.

Cada participante debía poner su predicción en un buzón, aquel que más se acercarse se la llevaría a la casa.

Su hipótesis era que sólo podría ser bueno aquel que estuviese formado y tuviese experiencia en el rubro, el resto serían muy malos, tanto que aún promediando todas sus predicciones estarían muy lejos del valor real y del participante “sabio”

Sin embargo, sucedio todo lo contrario. Cuando se promediaron todas las observaciones se obbutvo un valor muy cercano al peso real.

Esto pasa porque un monton de estimadores mediocres o malos promediados pueden ser un muy buen estimador → “inteligencia colectiva”

Cada una de las personas funciona como un modelo que en principio no hace una buen prediccion. Algunos por arriba, otros por debajo del valor, y promediados dan una buena prediccion.

- Entrenar varios modelos, c/u sobre datos distintos.
- Cada modelo sobre-ajusta de manera diferente.
  - Cada modelo: bajo sesgo, alta varianza.
  - Por ejemplo: árboles profundos.

Bias (sesgo)

El error debido al Bias de un modelo es simplemente la diferencia entre el valor esperado del estimador (es decir, la predicción media del modelo) y el valor real.

$$ErrorBias = Valor\ esperado\ -\ Valor\ real$$

Cuando se dice que un modelo tiene un **bias muy alto** quiere decir que el modelo es muy simple y no se ha ajustado a los datos de entrenamiento (suele ser **underfitting**), por lo que produce un error alto en todas las muestras: entrenamiento, validación y test

El error de **sesgo (bias)** es un error de “suposiciones erróneas” en el algoritmo de aprendizaje.

Un **sesgo alto** puede hacer que un algoritmo pierda las relaciones relevantes entre las características dadas y los resultados esperados (underfitting)

Varianza

La varianza de un estimador es cuánto varía la predicción según los datos que utilicemos para el entrenamiento.

La varianza es un error de sensibilidad a pequeñas fluctuaciones en el conjunto de entrenamiento.

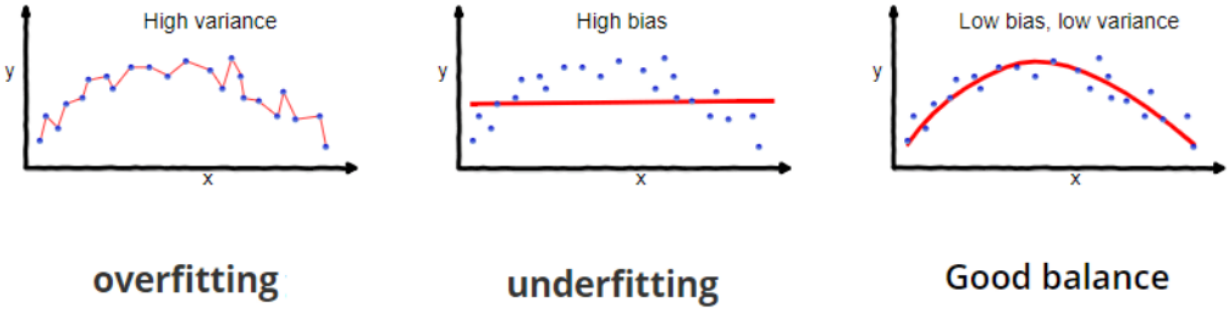
Un modelo con **varianza baja** indica que cambiar los datos de entrenamiento produce cambios pequeños en la estimación.

→ Da lo mismo entrenar con un conjunto, que con otro.

Al contrario, un modelo con **varianza alta** quiere decir que pequeños cambios en el dataset conlleva a grandes cambios en el output (suele ser **overfitting**).

→ Que cuando entrenamos un modelo, el modelo se aprende de memoria los datos que nosotros le damos. Se esta sobreajustando el modelo a los datos

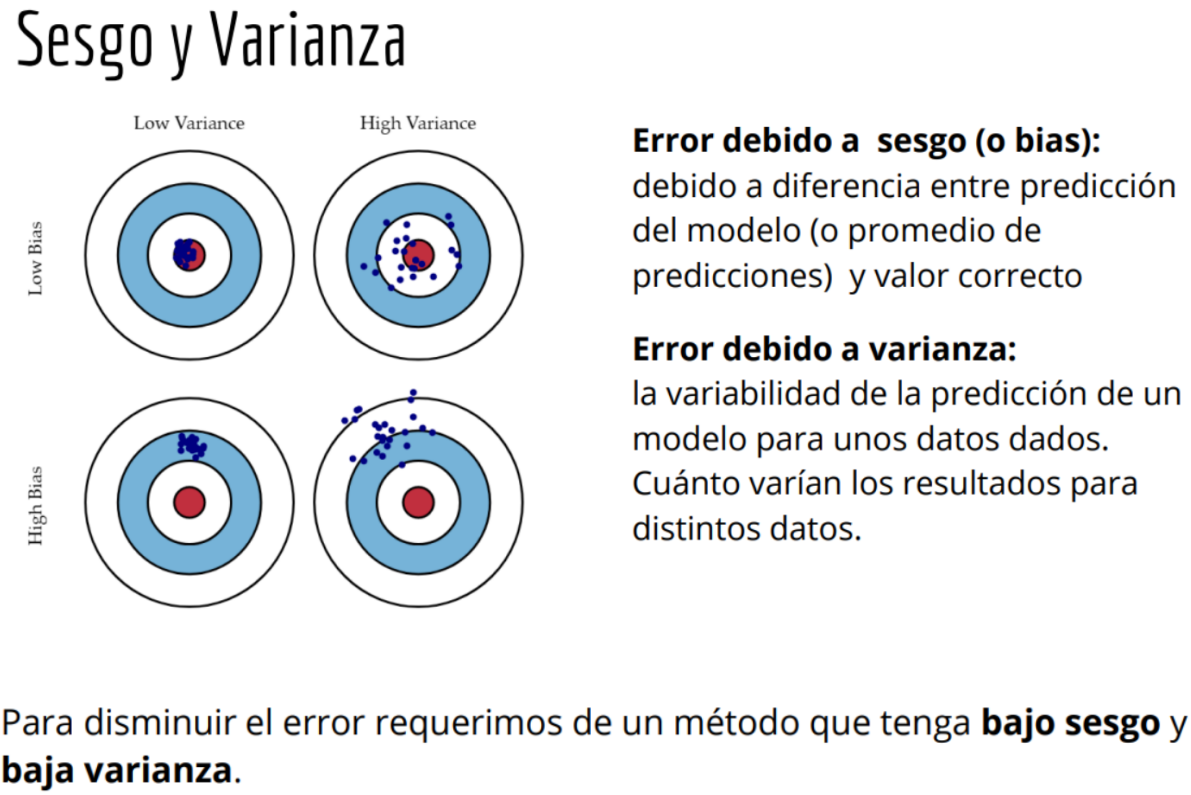
Una varianza alta puede resultar de un algoritmo que modela hasta el ruido aleatorio en los datos de entrenamiento (overfitting)



Overfitting: me aprendo los puntos de memoria. Se ajusta perfecto a los datos de entrenamiento. Pero al probar con los datos de prueba se tienen errores groseros. Al entrenar se tiene una presicion de 90% y al probar la presicion baja mucho, eso es porque esta sobreentrenado.

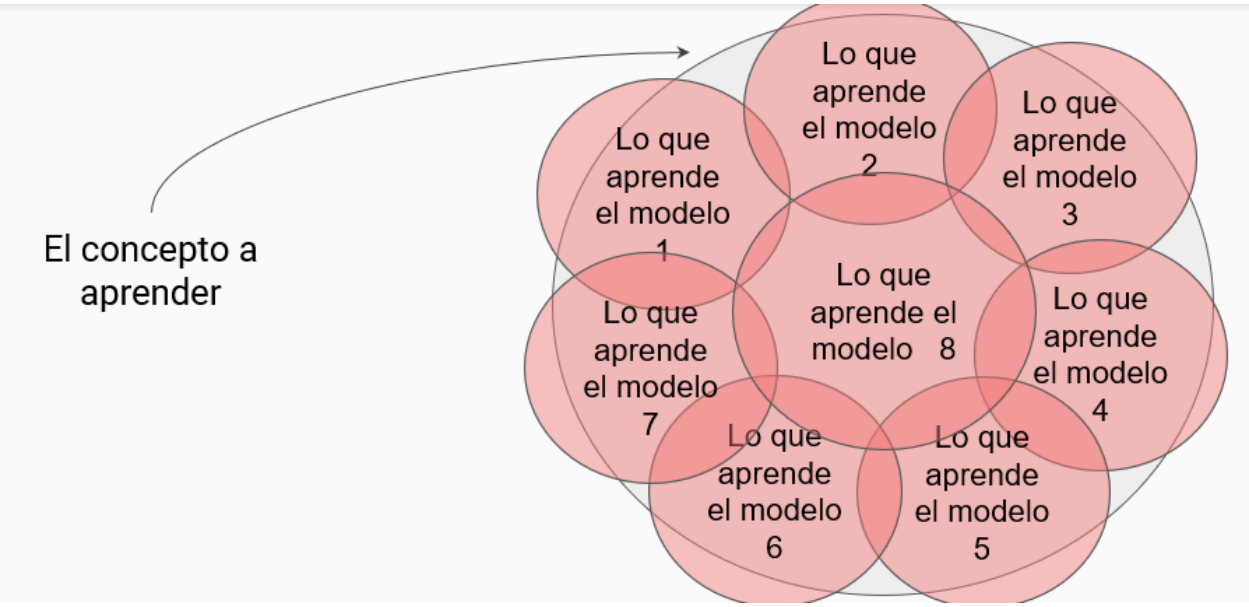
Underfitting: el modelo no termina de ajustar. Quizas porque no es demasiado complejo el modelo. Precide mal en todos lados (no como en overfitting que predice bien entrenamiento pero mal en test)

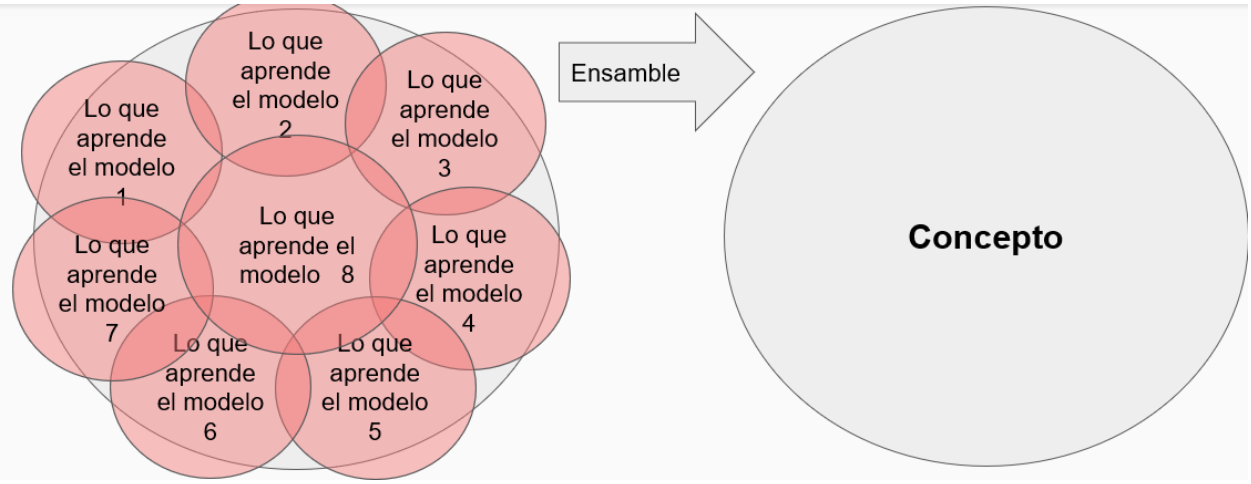
Lo ideal es tener un buen balance: bajo sesgo y baja varianza



Tener diferentes modelos, donde cada uno puede aprender una parte de ese concepto. De un set de datos complejo, puedo tener muchos modelos que aprendan pequeñas

partes diferentes, entocnes cada uno puede aprender bien su parte de los datos. Ninguno puede predecir la totalidad, pero todos juntos son una buena aproximacion de ese concepto, de ese modleo global que quiero entrenar.





Que hacemos una vez que tenemos entrenados todos estos modelo?

Votación:

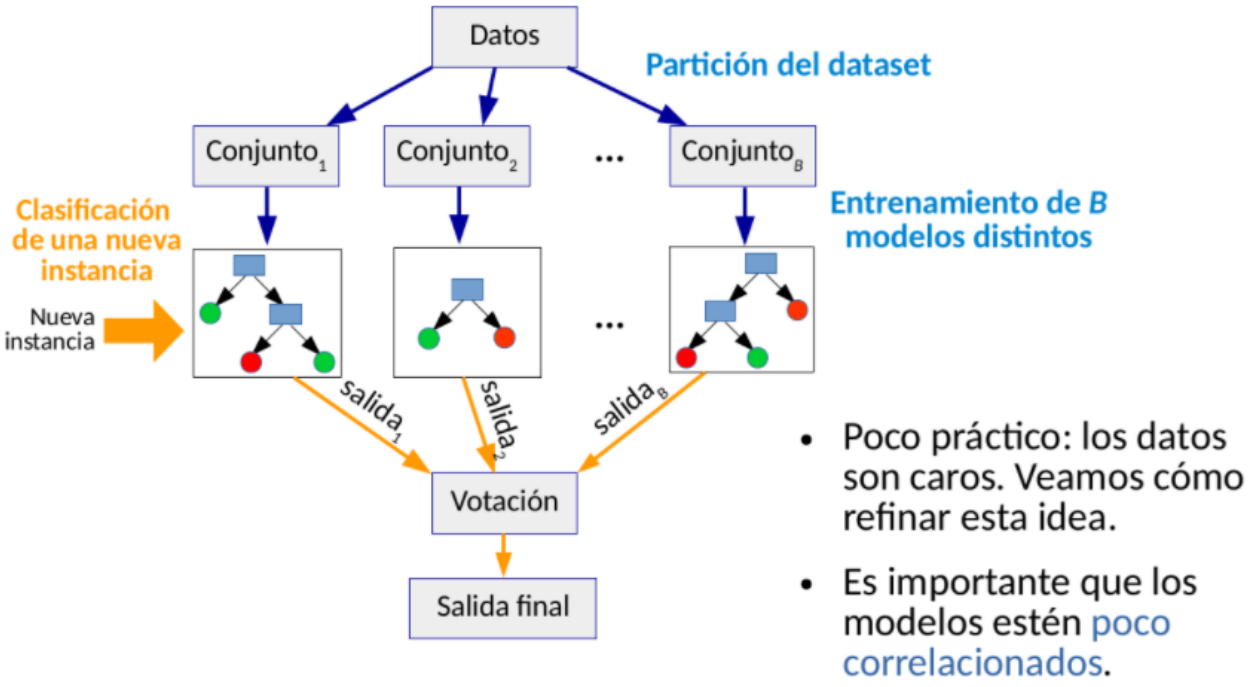
Para una nueva instancia, clasificarla con todos los modelos, y devolver la clase más elegida.

La votación reduce la varianza de la clasificación. (Random Forest)

Si los modelos individuales devuelven probabilidades, se puede hacer una votación ponderada.

Bagging

Tecnica que esta detras de Random Forest



1. Particion en n conjuntos de datos
2. Para cada conjunto entrenamos un modelo o clasificador, por ejemplo un arbol
3. Una vez que los arboles estan entrenados, cada uno va a dar una salida cuando esten ya en produccion
4. Se hace una votacion de la salida que mas salio.
5. La que sea mayoritaria es la que devuelve el modelo al final

Es costoso computacionalmente

Bootstrap: muestras aleatorias con reemplazo

Bagging es una técnica que consiste en construir nuevos conjuntos de entrenamiento usando bootstrap para entrenar distintos modelos, y luego combinarlos.

Bagging: paso a paso

1. Dividimos el conjunto de entrenamiento en distintos subconjuntos, obteniendo como resultado diferentes muestras aleatorias. (Boostrap)
    a. Las muestras son uniformes (misma cantidad de individuos)
    b. Son muestras con reemplazo (los individuos pueden repetirse en el mismo conjunto de datos)
2. Entrenamos un modelo con cada subconjunto
3. Construimos un único modelo predictivo a partir de los anteriores

Hasta aca es como funciona Random Forest

Ensamble homogeneo??

Características

- Disminuye la varianza en nuestro modelo final
- Muy efectivo en conjuntos de datos con varianza alta
- Puede reducir el overfitting
- Puede reducir el ruido de los outliers (porque no aparecen en todos los datasets)
- Puede mejorar levemente con el voto ponderado

Problemas al usarlo con árboles

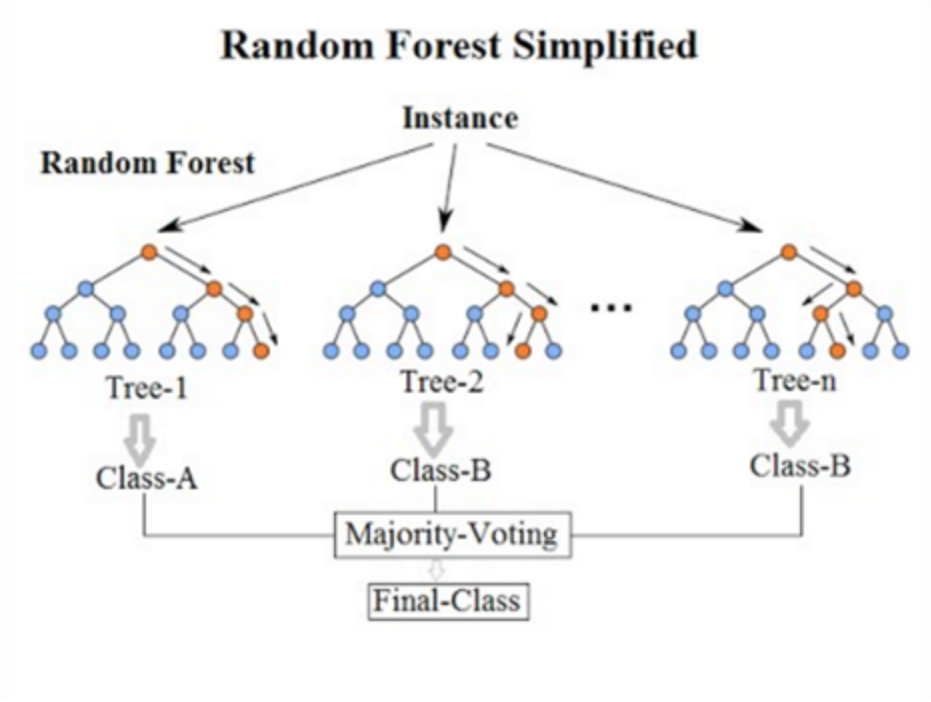
● Si pocos atributos son predictores fuertes, todos los árboles se van a parecer entre sí!

Ej: dataset con 50 columnas, pero solo 3 son la simportantes que me van a definir a grandes rasgos la clasificacion y la salida. En ese caso no tiene sentido usar bagging, porque todos los arboles van a tender a parecerse entre si y a utilizar los mismos atributos de la misma manera.

● Esos atributos terminarán cerca de la raíz, para todos los conjuntos generados con bootstrap

Bagging: Random Forest

Igual a bagging tradicional, pero en cada nodo, considerar sólo un subconjunto de m atributos (m columnas) elegidos al azar.

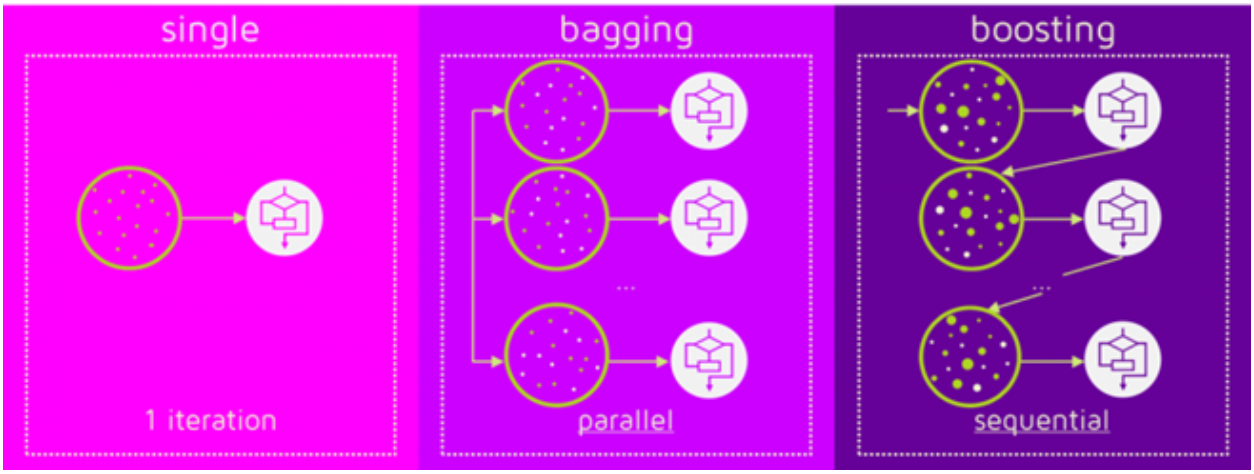


Boosting

Alternativa a Bagging.
Buscar nuevos modelos para las instancias mal clasificadas por los anteriores.

Con Bagging construimos todos los modelos de forma paralela, cada modelo es independiente de los demas.

En Boosting se entrenan los modelos de forma secuencial, a traves de un ciclo de iteraciones.

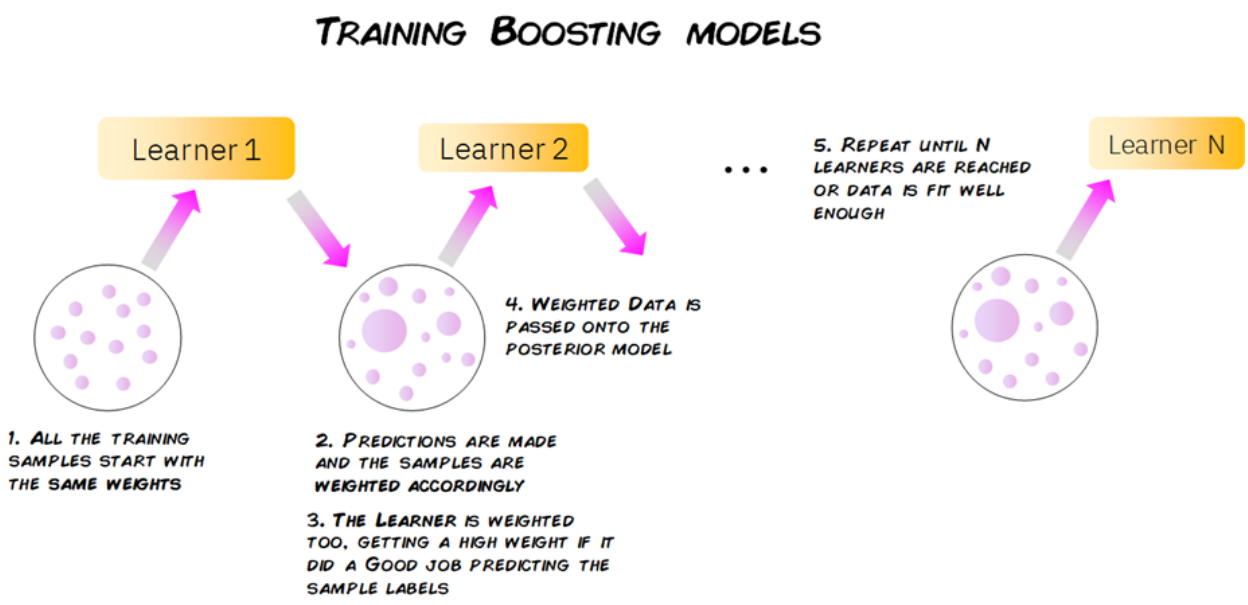


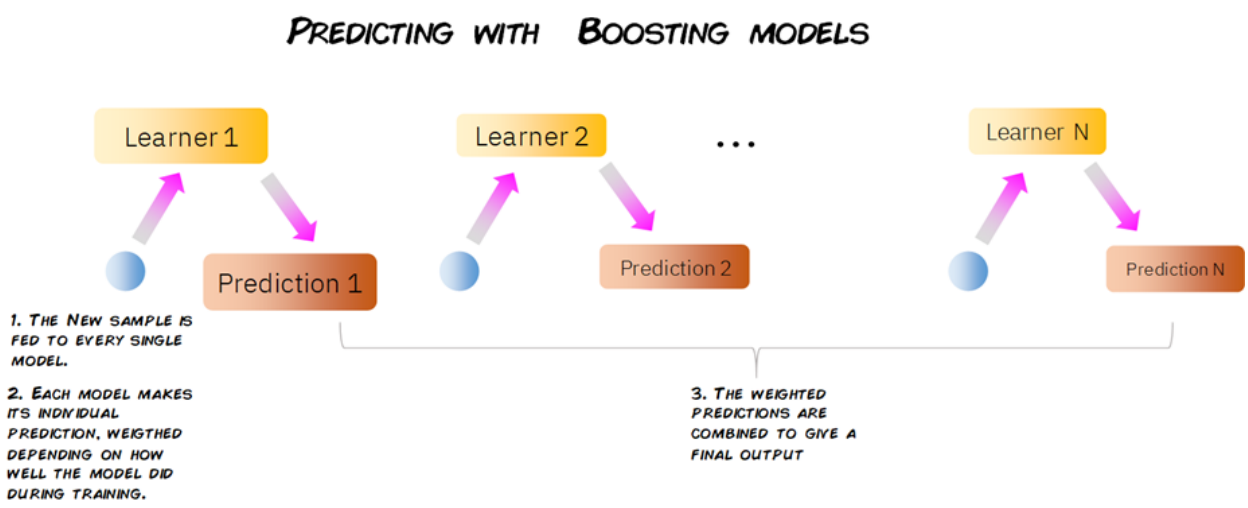
1. Comenzar con un modelo (simple) entrenado sobre todos los datos: h_0

2. En cada iteración i, entrenar h_i dando mayor importancia a los datos mal clasificados por las iteraciones anteriores.

3. Terminar al conseguir cierto cubrimiento, o luego de un número de iteraciones.

4. Clasificar nuevas instancias usando una votación ponderada de todos los modelos construidos.





Boosting: Resumen

- Necesita pesos, esto implica que:
  - Debemos adaptar algoritmo de aprendizaje
  - Y tomar muestras con reemplazo según pesos
- Puede sobreajustar

Si yo voy entrenando cada vez un modelo para que ajuste los errores del modelo anterior, podria termminar rapidamente en un sobreajuste y aprendiendo de memoria

todos los dtaos de entrenamiento.
Para evitar esto XGBoost va a implementar ciertos hiperparametros de regularizacion, para que estime mal, para convertirlo en un peor estimador.

Modelos

- AdaBoost
- Gradient Boosting
- XGBoost: eXtreme Gradient Boosting

- La velocidad de entrenamiento de XGBoost es mucho menor gracias a su implementación y a estar mejor orientado al uso eficiente del hardware (GPU)
- El accuracy (o la métrica adecuada) también es mejor debido a que XGBoost maneja mejor el overfitting mediante regularizaciones