Maestría en Explotación de Datos y Descubrimiento del Conocimiento

Unidad II, Clase 7

Aprendizaje Automático

Ensambles

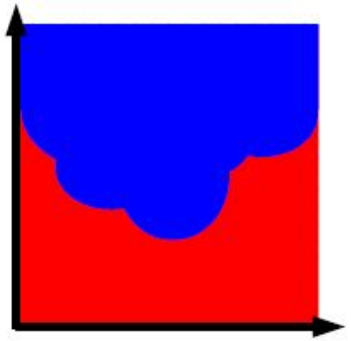
Laura de la Fuente, Hernán Bocaccio
Ayudantes: Gastón Bujía, Diego Onna y Sofía Morena del Pozo

Dirección de e-mail de la materia:
datawillconfess@gmail.com

Itinerario de la clase

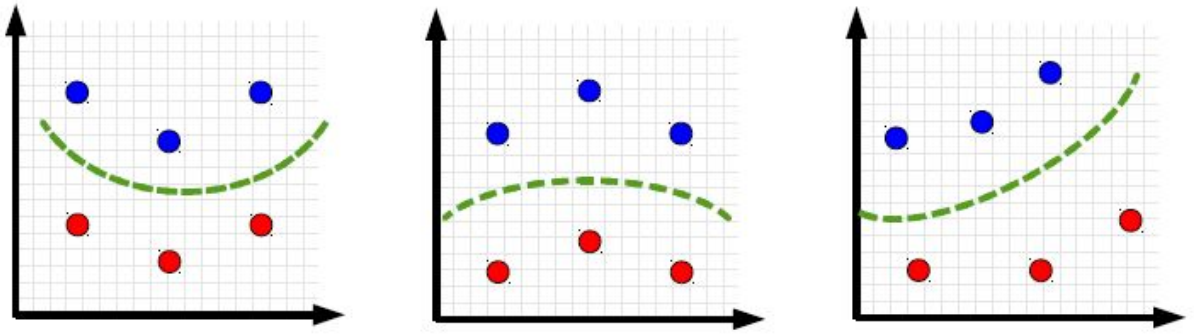
- Trade-off sesgo/varianza
- Complejidad de los modelos
- La maldición de la dimensionalidad
- Reducción de la dimensionalidad
- Ensambles de modelos
- Bagging
- Random Forest
- Boosting
- Gradient Boosting
- Stacking

Trade-off sesgo/varianza



Función objetivo (desconocida)

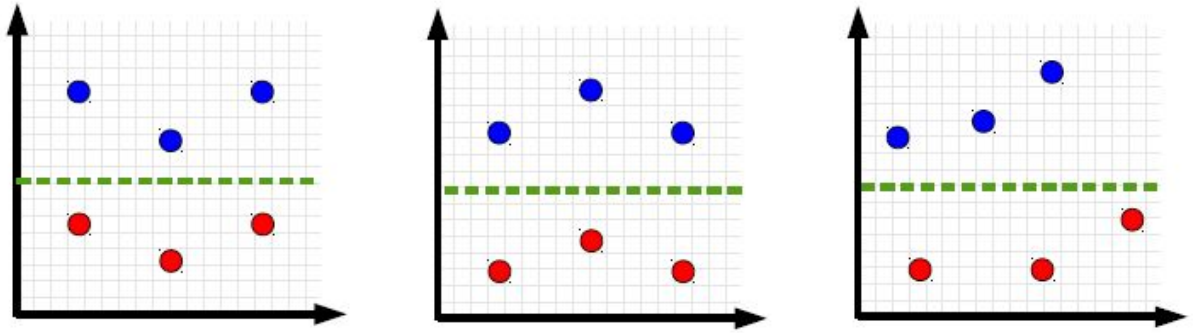
Posibles conjuntos de datos de entrenamiento



<u>Hipótesis: parábola</u>

Bajo sesgo: más posible acercarse al objetivo

Alta varianza: los modelos construídos cambian mucho según la muestra



<u>Hipótesis: recta horizontal</u>

Alto sesgo: menos posible acercarse al objetivo

Baja varianza: los modelos construídos cambian poco según la muestra

Trade-off sesgo/varianza

Error esperado del modelo sobre distintos sets de datos $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_N\}$

$$\mathbb{E}_{\mathcal{D}}(MSE) = \{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}^2 + \mathbb{E}_{\mathcal{D}}\left[\{y(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})]\}^2\right] + \sigma^2$$

h(x) = E[t|x]

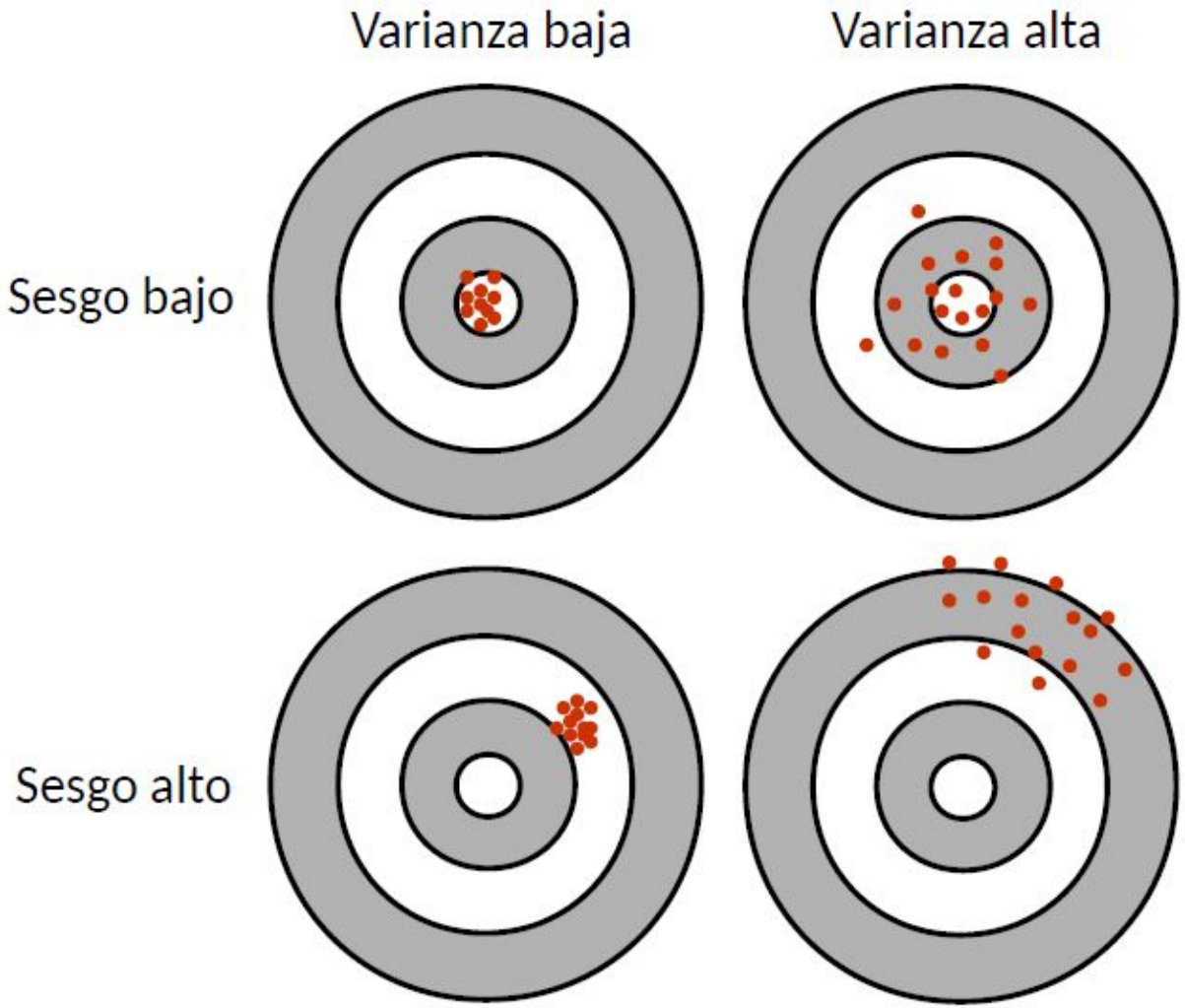
$$\mathbb{E}_{\mathcal{D}}(MSE) = \textbf{Bias}^2 + \textbf{Variance} + \sigma^2$$

Error debido a sesgo (o bias): diferencia entre predicción del modelo (o promedio de predicciones) y valor correcto.

Error debido a varianza: variabilidad de la predicción de un modelo para distintos datos dados.

Disminuir el error: buscamos un método que tenga **bajo sesgo y baja varianza**.

Trade-off sesgo/varianza

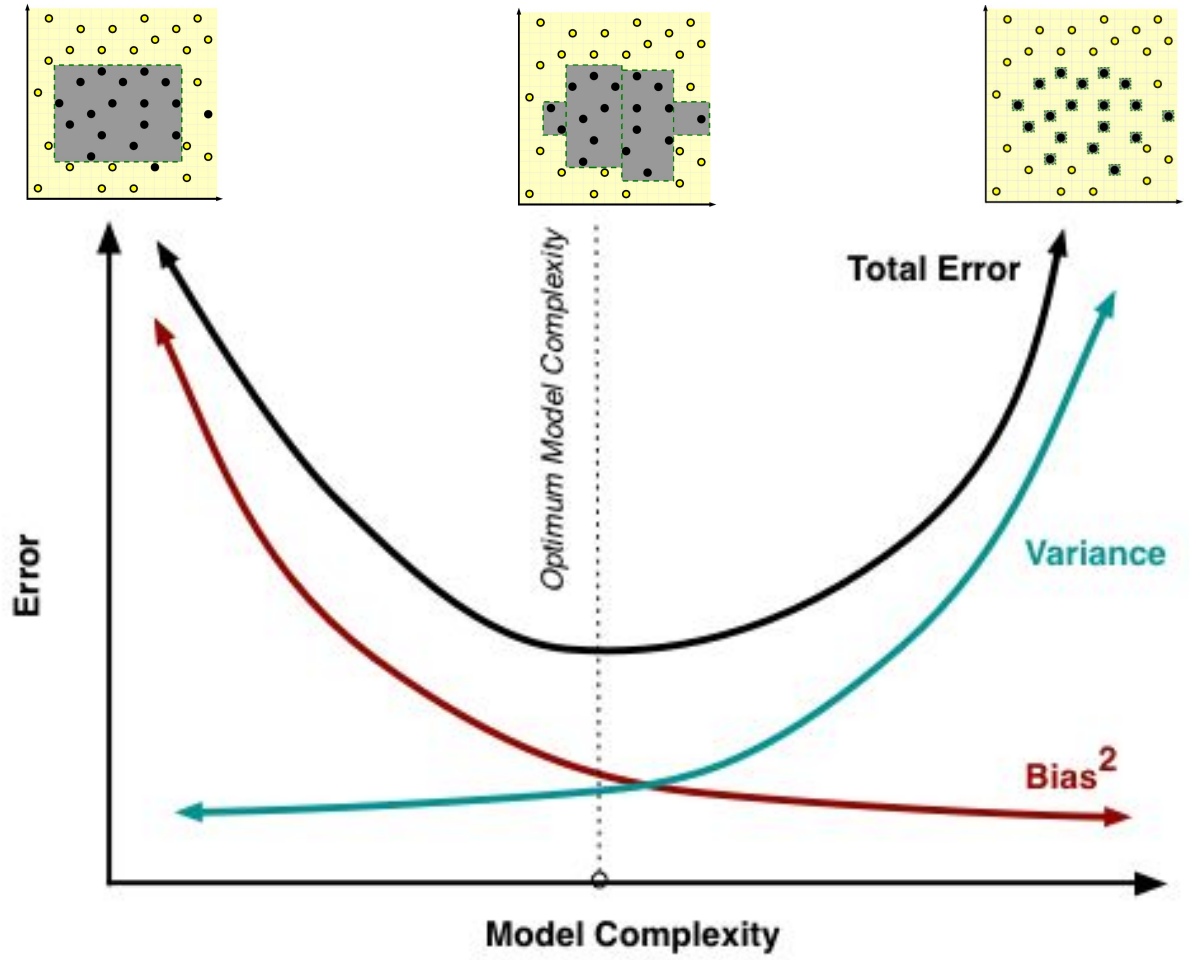


Escenarios posibles de desempeño de un modelo

Se muestran 4 situaciones distintas que diferencian 4 modelos

Los puntos representan el desempeño en distintos conjuntos de datos

Complejidad de los modelos



Modelo **más complejo = más parámetros**, o dependencias más complejas (**no lineales**)

Tener más parámetros lo hace más **flexible**, i.e. poder **ajustar mejor a los datos**

Puede llevar al **problema de sobreajuste**

El modelo se **ajusta al ruido** de un conjunto específico de **datos** (**bajo sesgo**)

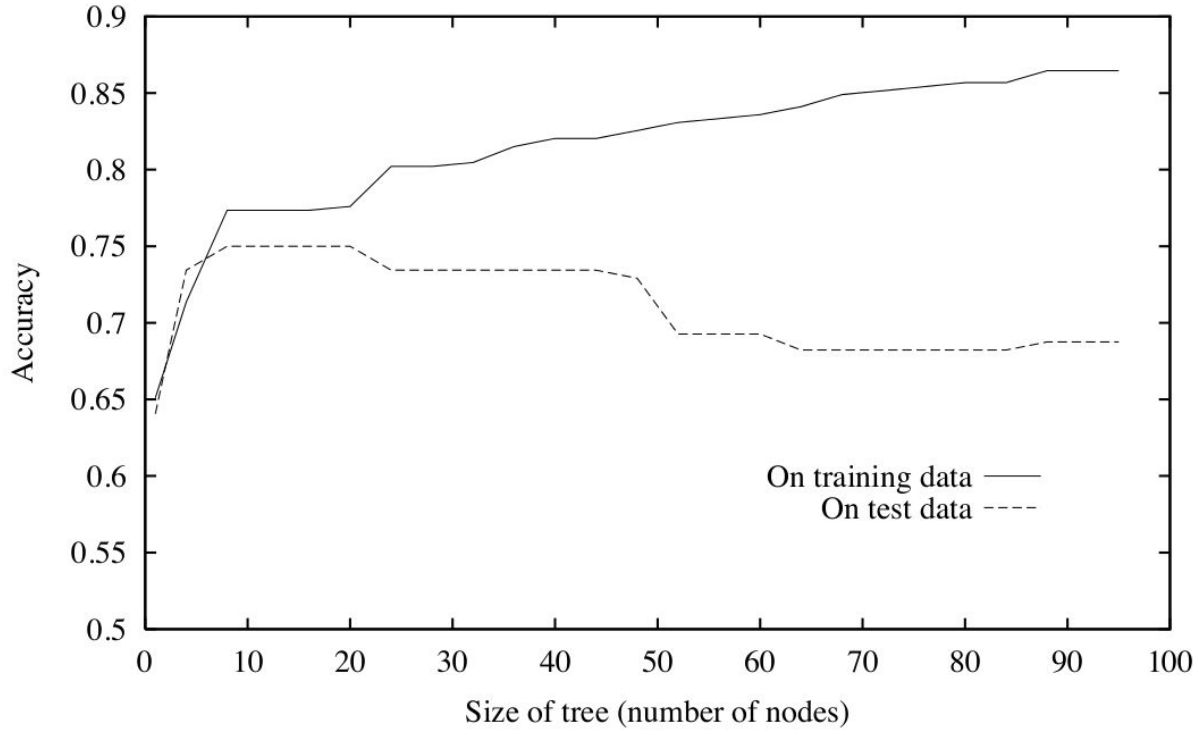
Cambia mucho si uso **otros datos** (**alta varianza**) por lo que **no generaliza** bien

Complejidad de los modelos

Complejidad y flexibilidad

Conociendo un modelo podemos decir que es más complejo que otro (parámetros, dependencia)

Pero cómo defino cuándo un modelo es “muy” complejo?



Decimos que es “muy” complejo cuando el modelo sobreajusta y no generaliza. Podemos ver las **curvas de complejidad**

<u>Posible solución</u>:
Fine-tuning de los **hiperparámetros** que **optimizan la complejidad** del modelo a partir de evaluar en datos no usados en el entrenamiento (**validación**). Esto es lo que llamamos **selección de modelos**

Complejidad de los modelos

Complejidad y flexibilidad

Conociendo un modelo podemos decir que es más complejo que otro (parámetros, dependencia)

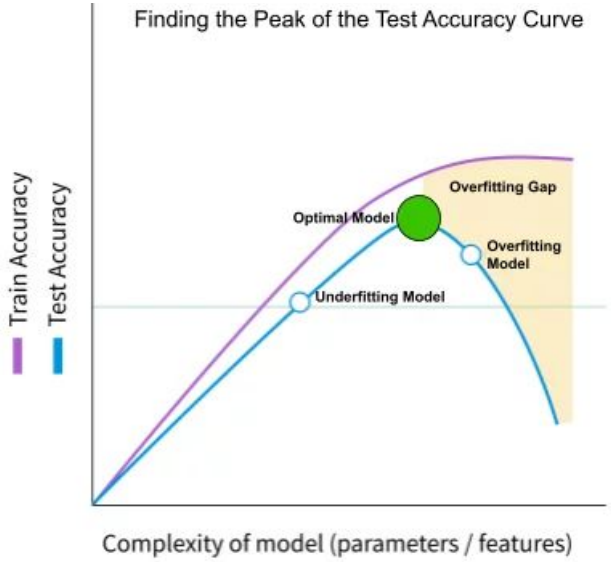
Pero cómo defino cuándo un modelo es “muy” complejo?

En general, lo importante no es si el modelo surge de un espacio de hipótesis con dependencias complejas o no, si no que lo que importa es si esa complejidad lo hace tan flexible respecto de los datos (sensible al ruido) que estoy sobreajustando (no generalizo)

No sólo que no depende de una cantidad absoluta de parámetros del modelo, si no que puede también depender de los datos

- Estructura
  - Complejidad de la función objetivo
- Ruido, outliers
- Cantidad de datos
- Cantidad de atributos

} **Relación**

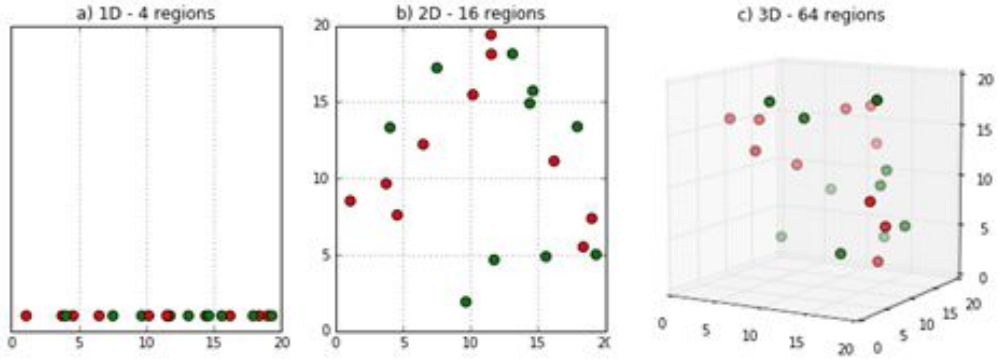


La maldición de la dimensionalidad

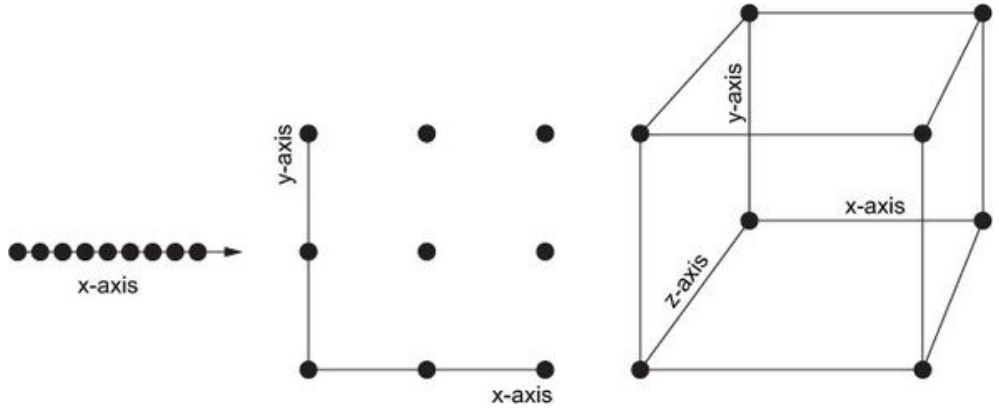
Cuantos más parámetros tengo, más grande es el espacio de parámetros que tengo que ajustar con el modelo

Los datos están más **dispersos**, son menos representativos del espacio (parámetros).

Hay regiones en las que no tengo datos.



Las **distancias** entre los datos **se empiezan a parecer cada vez más**, tienen menos relevancia para diferenciar datos.



La cantidad de datos que necesito crece exponencialmente

La maldición de la dimensionalidad

Problemas de la alta dimensionalidad

Distancia media entre puntos es grande.

Por lo tanto, datasets esparsos. Imposible obtener suficientes instancias para que los puntos se encuentren "cerca".

Las predicciones de los modelos son menos confiables, porque se parecen más a extrapolaciones.

Además las distancias tienden a parecerse. Difícil diferenciar.

Problemas prácticos:

- Entrenamientos costosos
- Manipulación de grandes datasets
  - En memoria
  - En disco

La maldición de la dimensionalidad

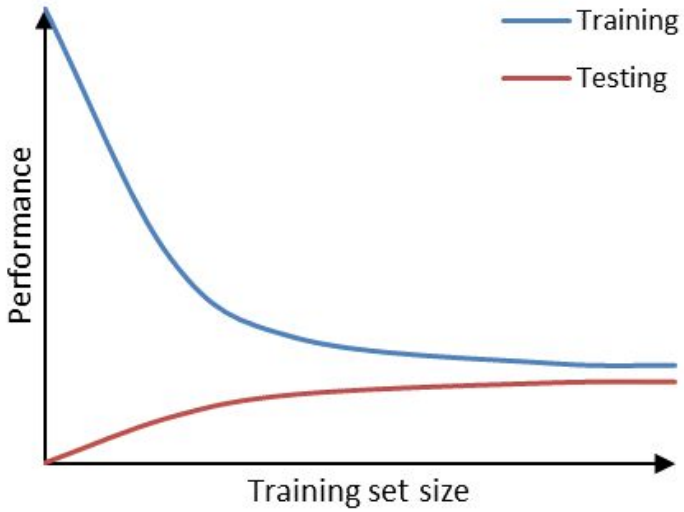
Posibles formas de tratarla

- Tomar más datos
  Puede ser muy costoso o incluso imposible, y no siempre soluciona
- Selección de atributos (feature selection)
  Elegir un subconjunto de atributos con algún criterio de relevancia (ingeniería de features; estadística; …)
- Reducción de la dimensionalidad
  Pasar del espacio N-dimensional de atributos a un espacio de menor dimensionalidad (PCA; LDA; MDS; t-SNE; …)
- Generar datos sintéticamente (data augmentation)
  Conservar estructura relevante de los datos y del sistema que estoy tratando (importancia de que no sea sólo data-driven)
- Regularización
  Penalizar pesos para reducir la flexibilidad del modelo y hacerlo menos sensible al ruido de los datos de entrenamiento
- Ensambles (... próximamente)
- …

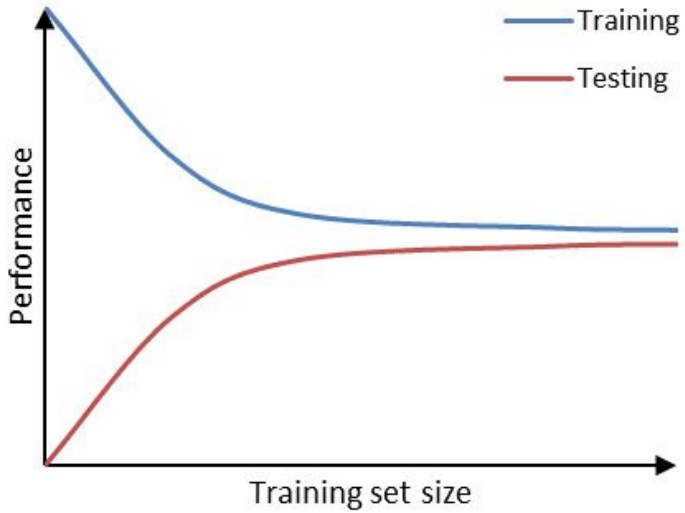
Tomar más datos

Curvas de aprendizaje

sesgo alto, varianza baja

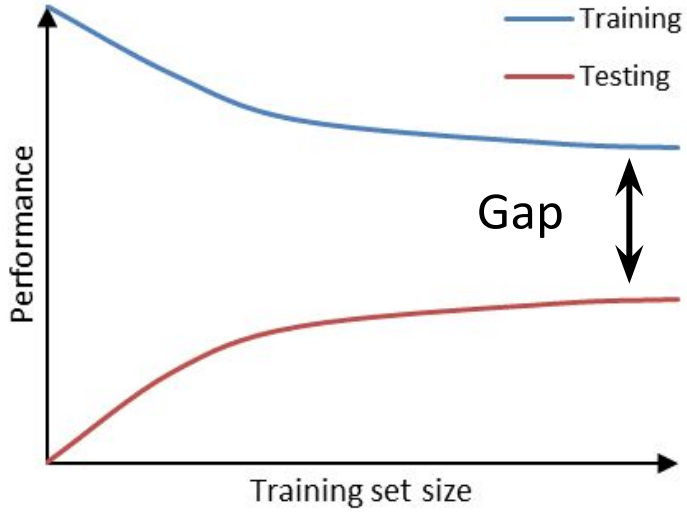


Si tengo sesgo alto, no importa cuantos datos agregue, el modelo no ajusta bien a los datos de entrenamiento.



Óptimo

sesgo bajo, varianza alta

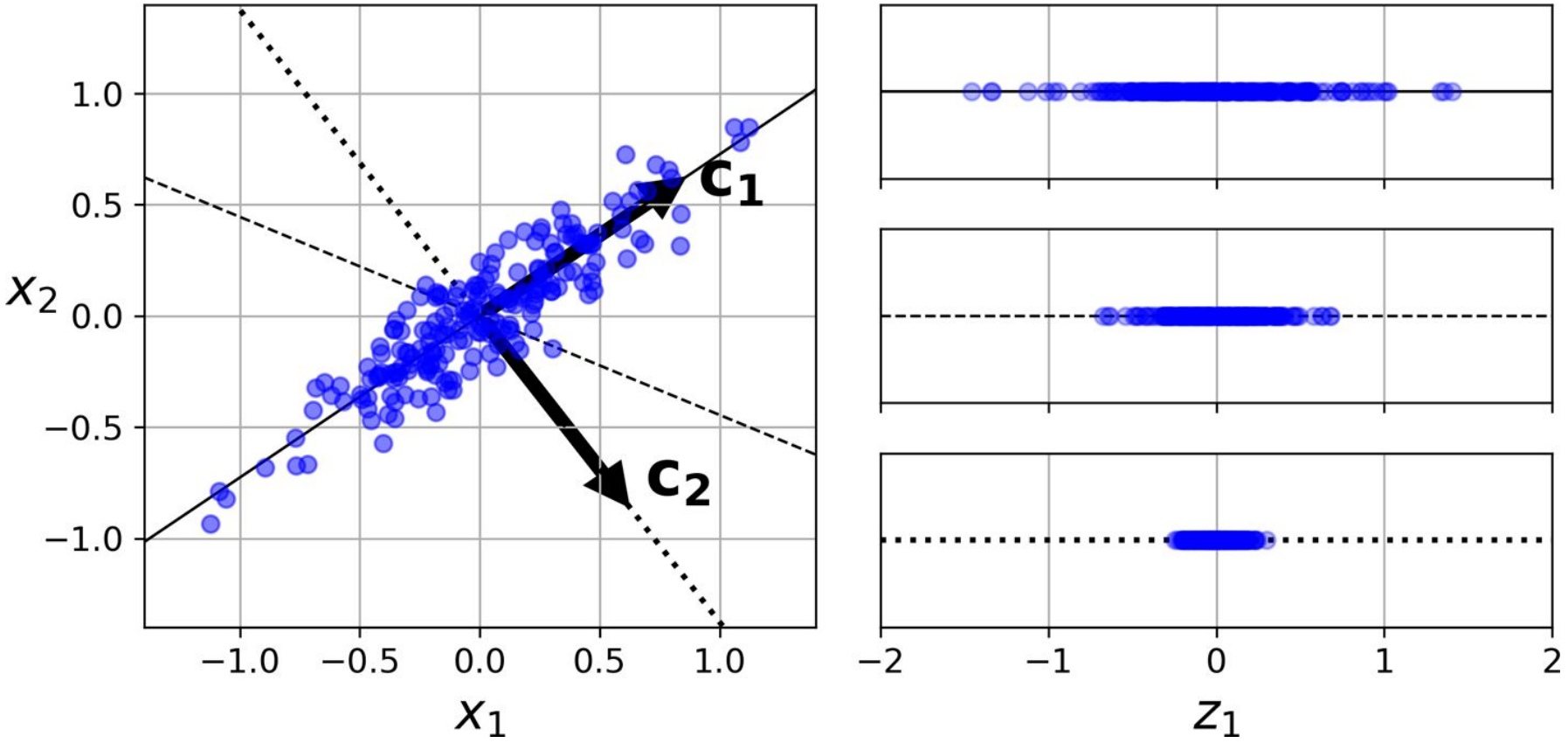


Si tengo varianza alta, no importa cuantos datos agregue, el modelo no generaliza bien.

Reducción de la dimensionalidad

Principal Component Analysis (PCA)

Rotar datos para encontrar las proyecciones ortogonales que maximizan la varianza (SVD)

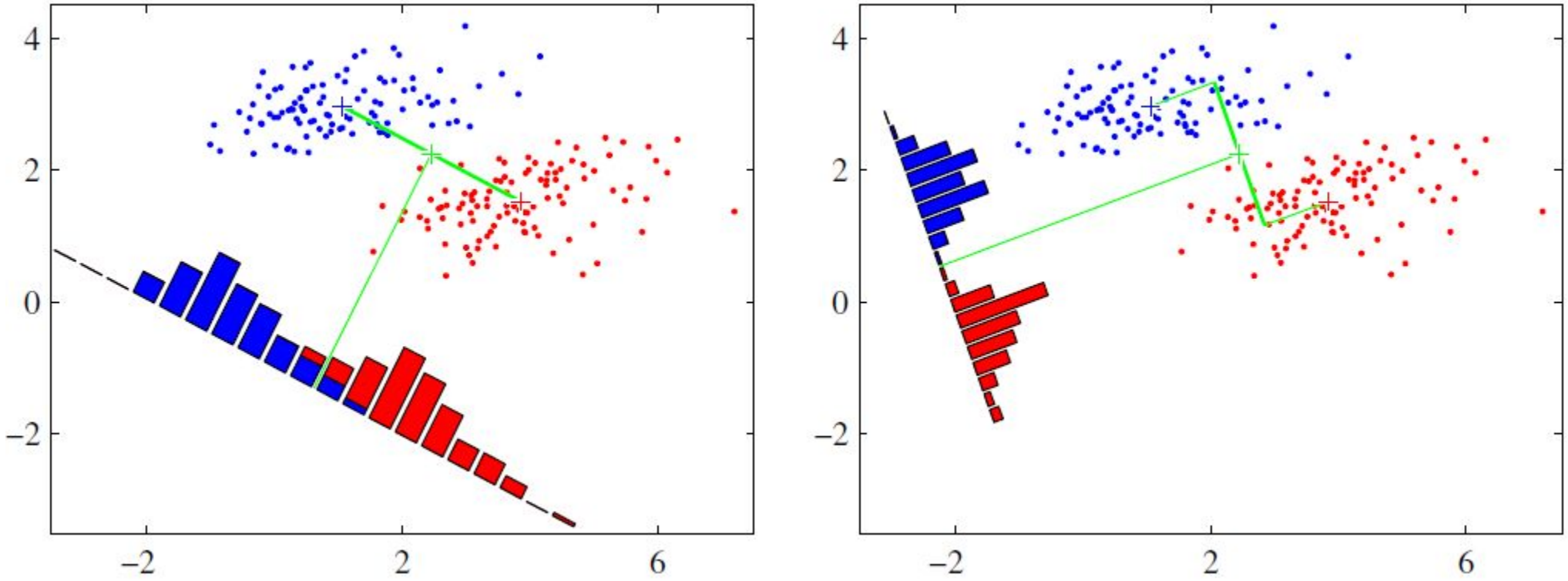


Géron 2019

Reducción de la dimensionalidad

Linear Discriminant Analysis (LDA)

Rotar datos para encontrar las proyecciones que maximizan la distancia entre centroides

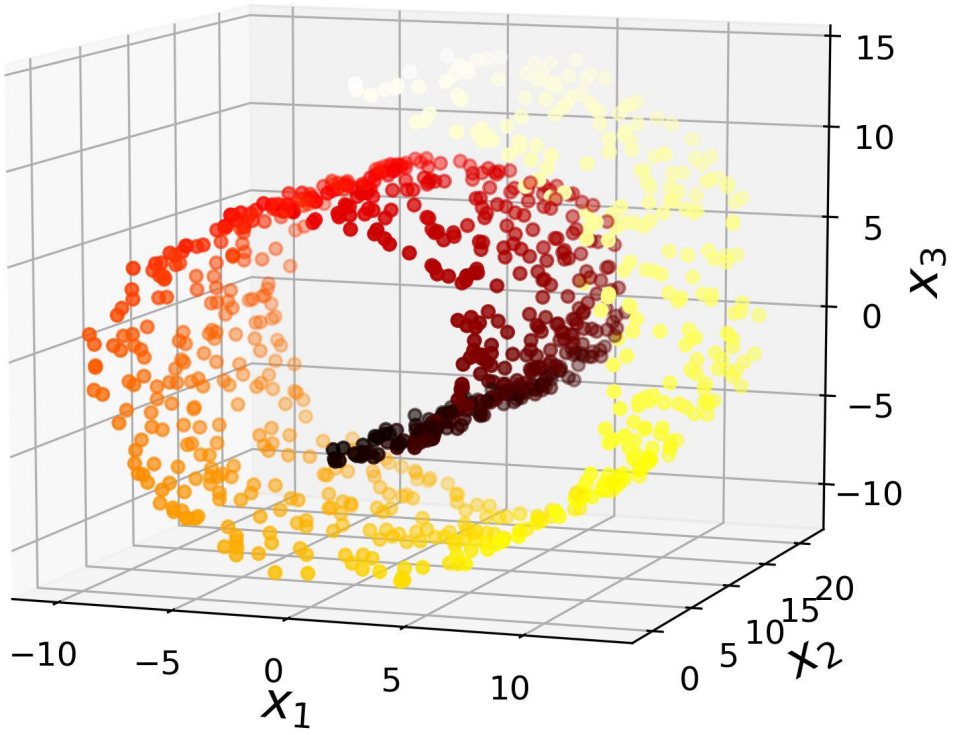


Bishop 2006

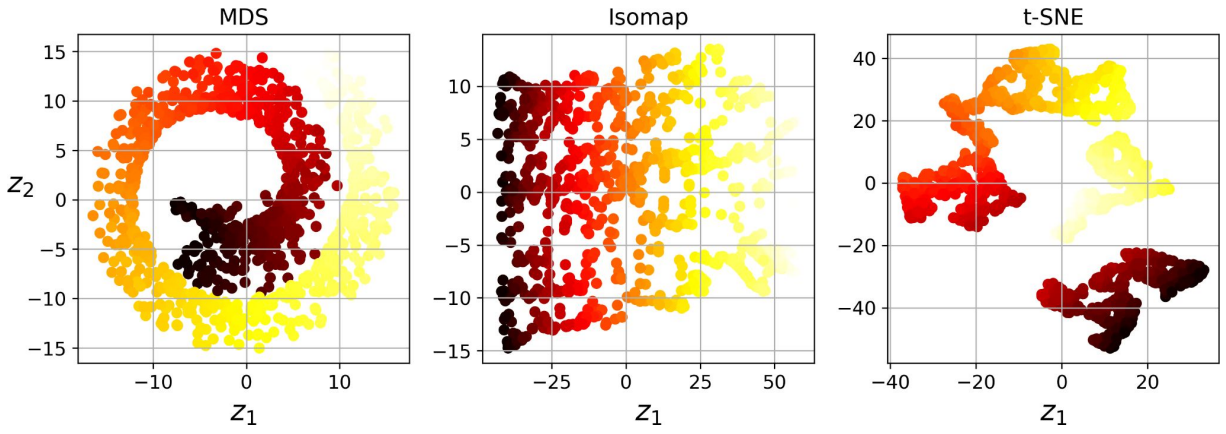
Reducción de la dimensionalidad

Manifold Learning*

Proyectar datos en dimensiones más bajas (embeddings), conservando algunas relaciones presentes en el espacio original



- Multidimensional Scaling (MDS): distancias
- Isomap: distancia geodésica entre vecinos cercanos (desenrolla)
- t-SNE: acerca instancias por afinidad



* sklearn.manifold

Géron 2019

Ensambles

Combinar varios modelos para hacer la predicción (mejorada)

- Voting (votación mayoritaria)
- Bagging o Bootstrap Aggregating
  - Random Forests
- Boosting
  - AdaBoost
  - GradientBoosting
- Stacking o Blending
- …

Combinando un conjunto de modelos construyo un **meta-modelo**

Tomo una **decisión conjunta** (e.g. votación, votación ponderada, promedio para regresión)

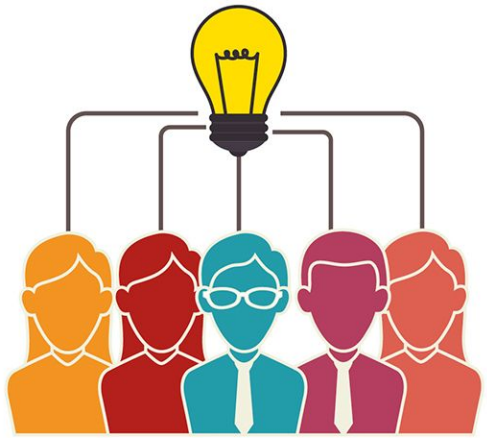
El todo es mayor a la suma de las partes

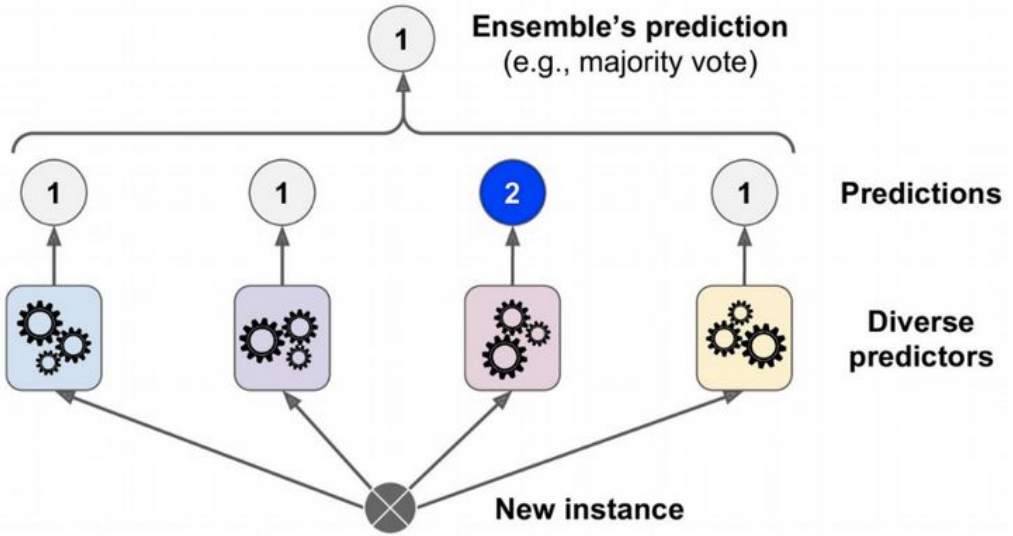
Ensambles

Combinación de modelos en paralelo

- Voting (votación mayoritaria)
- Bagging o Bootstrap Aggregating
- Random Forests
- Parte de Stacking o Blending
- …

$$Var(\overline{X}) = \frac{\sigma^2}{n}$$





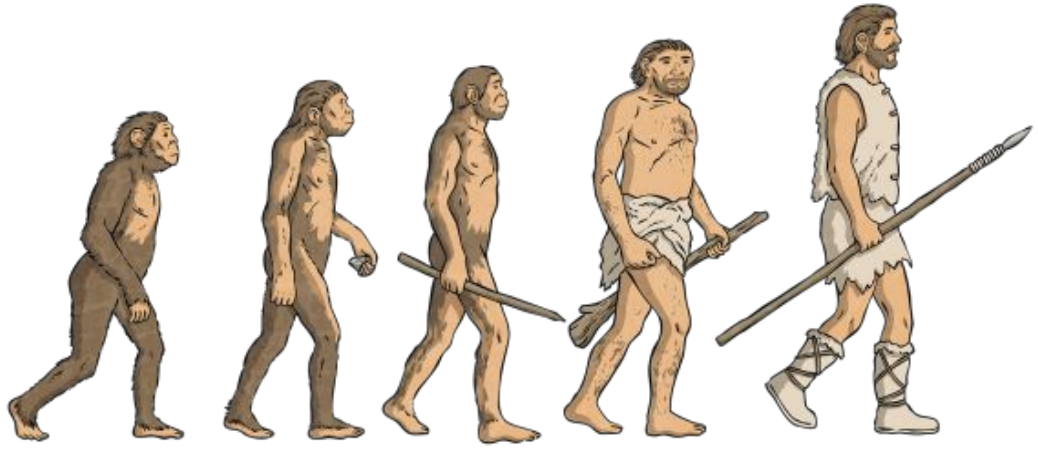
+ Usar **modelos fuertes**, de **bajo sesgo y alta varianza** (e.g. árboles profundos), y la votación mayoritaria (paralelo) **reduce la varianza**

+ Agregar modelos no sobreajusta

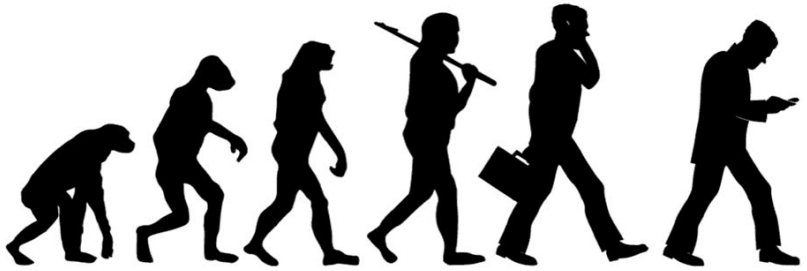
Ensambles

Combinación de modelos en serie

- Boosting
  - AdaBoost
  - GradientBoosting
- Parte de Stacking o Blending
- …



Es greedy (optimización local); difícil de paralelizar; ajustar el learning rate puede ser muy importante para balancear la convergencia de una secuencia de modelos naives sin sobreajustar



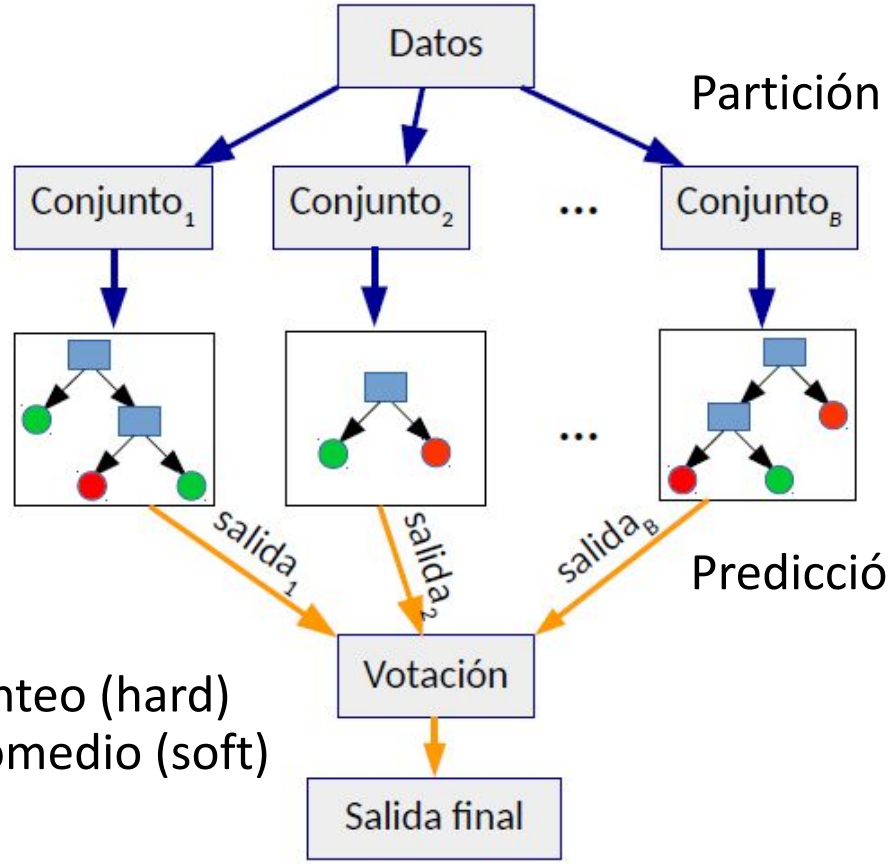
\+ Usar **modelos débiles**, de **alto sesgo** y **baja varianza** (e.g. árboles cortos), y usar información previa (secuencia) **reduce el sesgo**

\- Agregar modelos puede sobreajustar

Ensambles

Voting

Entrenamos modelos distintos y hacemos la predicción de acuerdo a la votación mayoritaria sobre un dato nuevo



Partición del dataset

Entrenamiento de B modelos distintos

Predicción en un dato nuevo

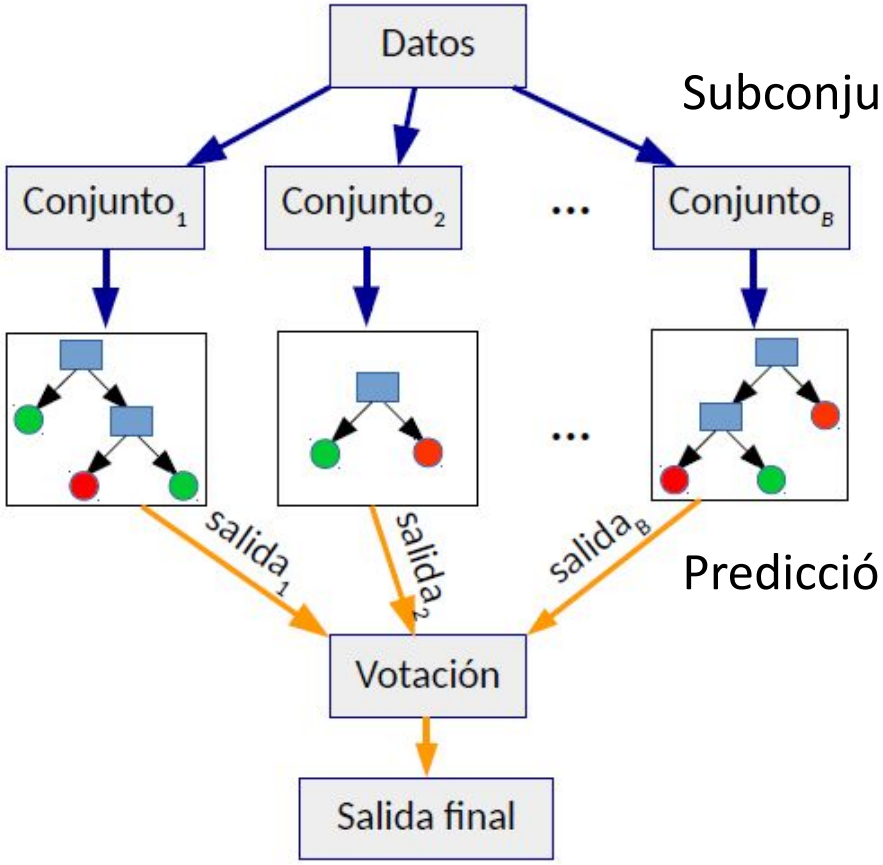
Conteo (hard)
P promedio (soft)

- Requiere de muchos datos
- Los modelos deben ser independientes

Ensambles

Bagging (**B**ootstrap **Agg**regat**ing**)

Construir nuevos conjuntos de entrenamiento usando bootstrap (muestreo con reemplazo de las instancias)



Subconjuntos de datos generados mediante bootstrap

Entrenamiento de B modelos distintos

Predicción en un dato nuevo

+ Requiere menos datos

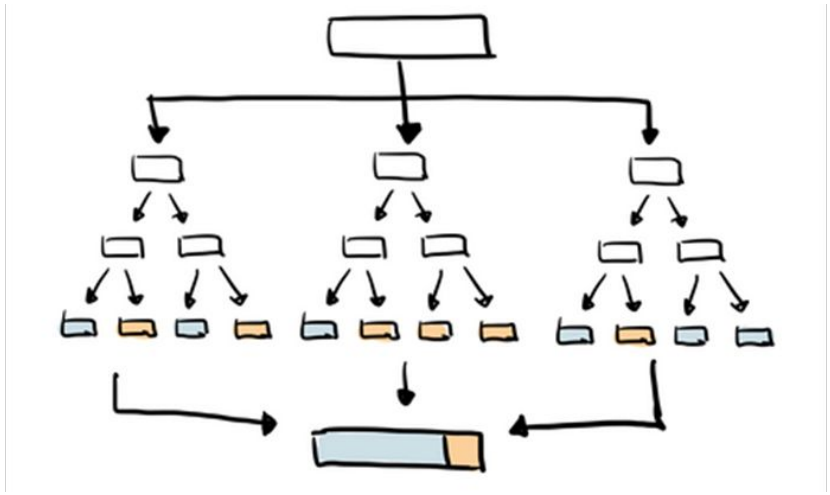
- Los modelos deben ser independientes

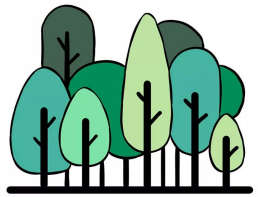
Ensambles

Random Forest

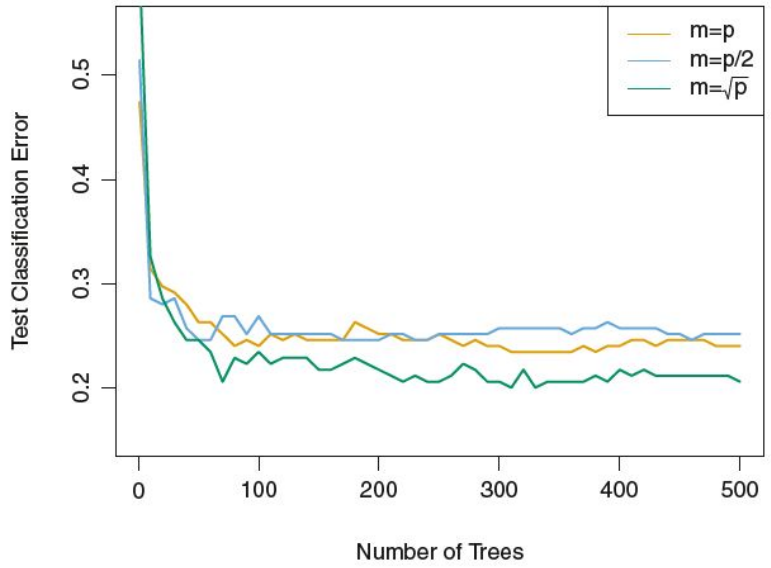
El problema de bagging con árboles es que los modelos no son independientes:

- Si pocos atributos son predictores fuertes, todos los árboles se van a parecer entre sí
- Esos atributos terminarán cerca de la raíz, para todos los subconjuntos generados





Random Forest es igual a **bagging**, pero en cada nodo, uso sólo un **subconjunto de atributos** (m) elegidos **al azar** en lugar de usar el total de atributos (p)



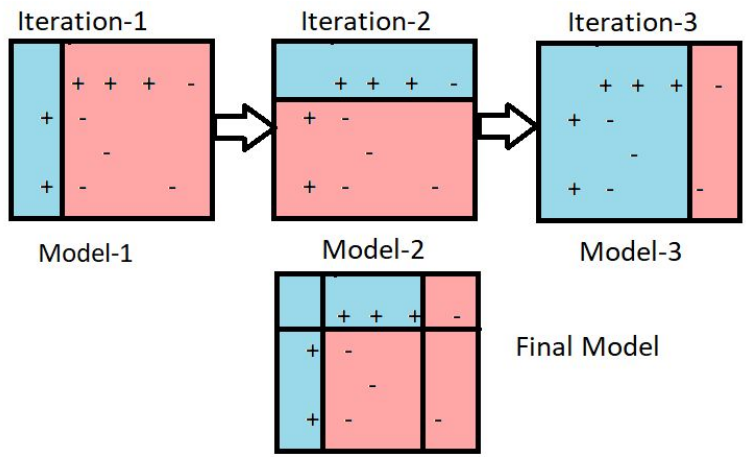
+ Requiere menos datos
+ Los modelos son independientes

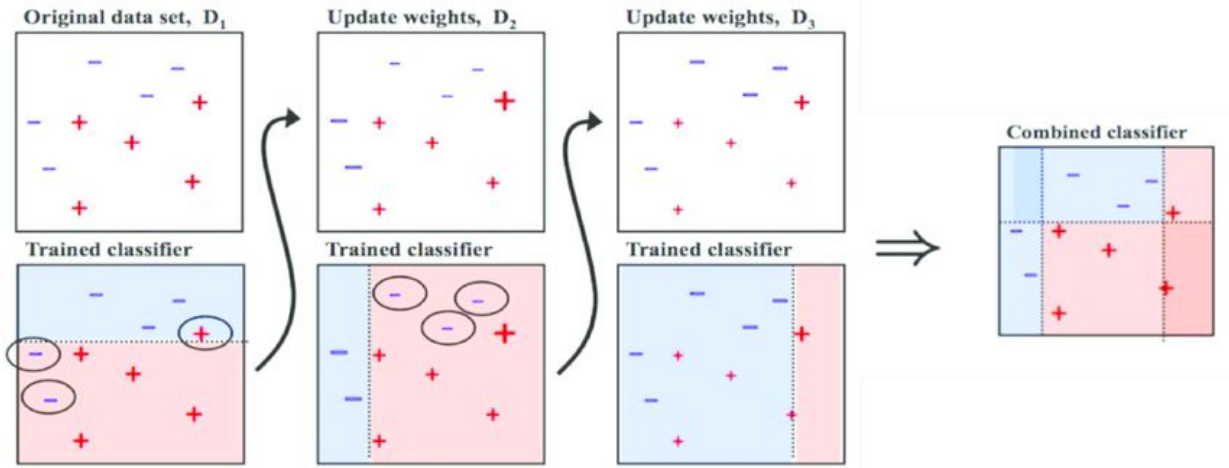
Ensambles

Boosting

Combinamos de manera secuencial muchos predictores, cada uno corrigiendo de alguna manera lo que hizo el anterior

- Comenzar con un modelo (simple) entrenado sobre todos los datos: h0
- En cada iteración i, entrenar hi dando mayor importancia a los datos mal clasificados por las iteraciones anteriores
- Terminar al conseguir cierto cubrimiento, o luego de un número de iteraciones
- Clasificar nuevas instancias usando una votación ponderada de todos los modelos construidos
- Los dos más populares son AdaBoosting y GradientBoosting

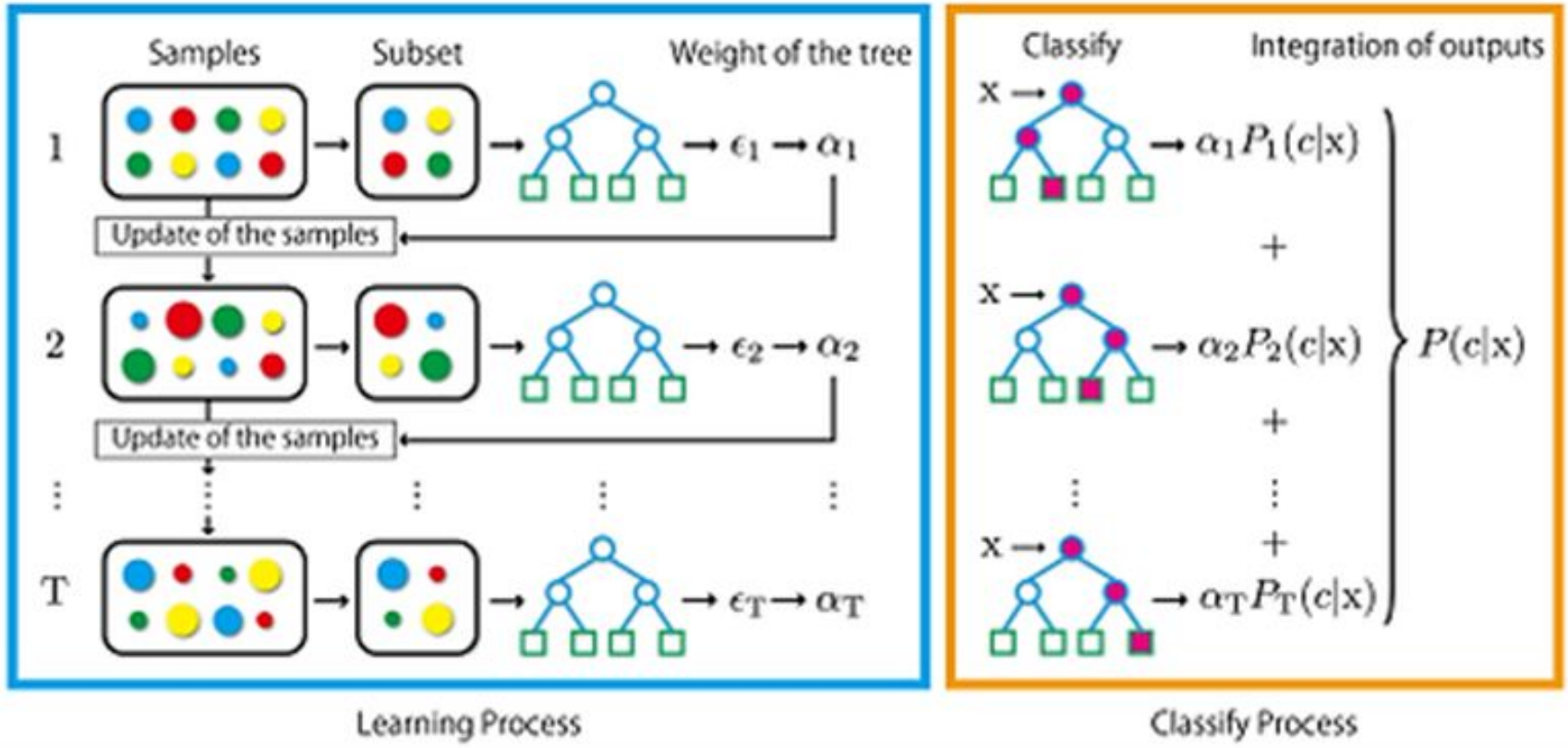




Ensambles

AdaBoost

Cada nuevo modelo usa los errores del modelo anterior para aplicar pesos a las instancias de manera de reducir el error de sesgo a cada paso

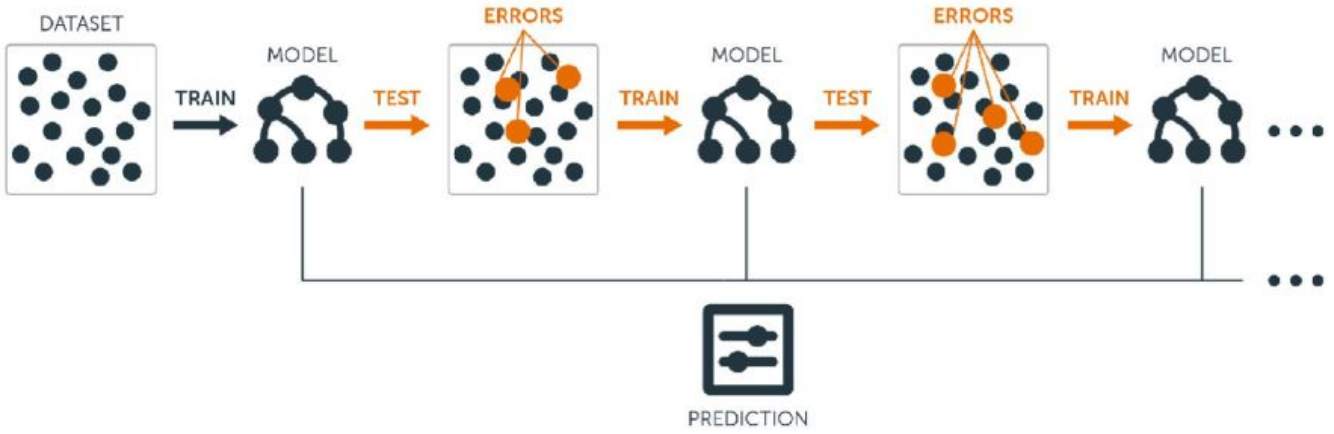


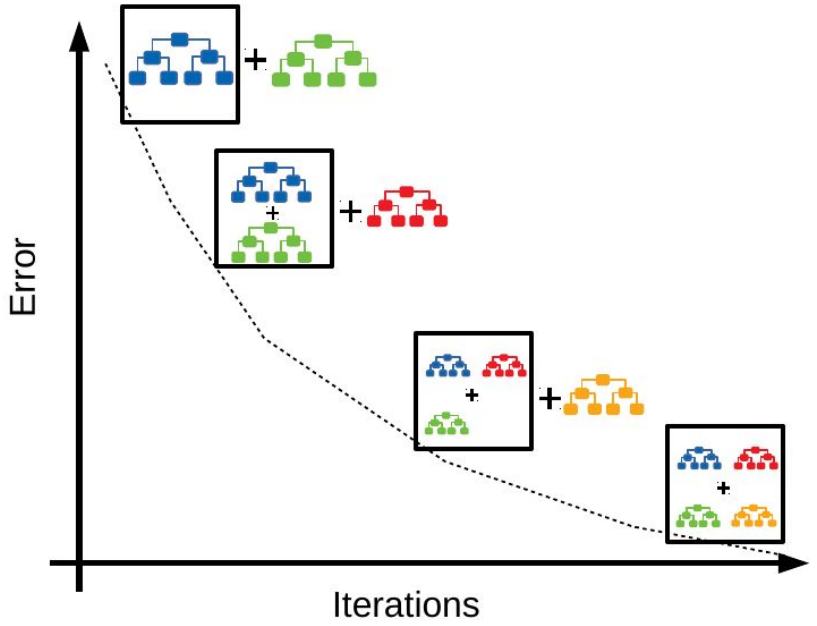
Ensambles

GradientBoosting

Cada nuevo modelo busca minimizar la función costo del modelo anterior de manera de reducir el error de sesgo a cada paso.

La diferencia con AdaBoost es principalmente de implementación, pero conceptualmente y a fines prácticos son muy similares (AdaBoost también reduce función costo).

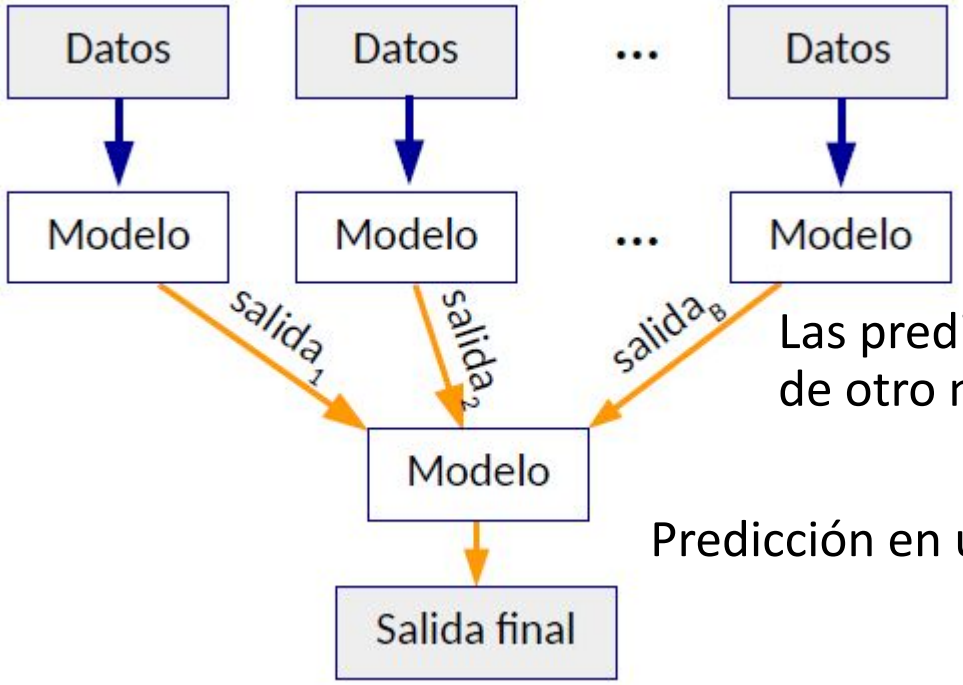




Ensambles

Stacking

Entrenar diferentes modelos (modelos base) y un modelo más, que decide, dada una instancia nueva, qué modelo usar. Los modelos intermedios aprenden una parte del problema. El modelo final usa una combinación de predicciones intermedias para la resolución general.



Mismos datos

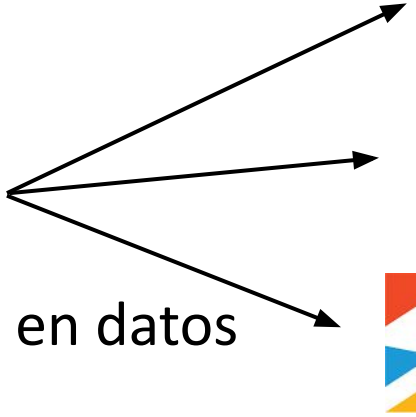
Entrenamiento de modelos distintos

Las predicciones de modelos base son la entrada de otro modelo que decide

Predicción en un dato nuevo

Ensambles

- Combino un conjunto de modelos para armar un meta-modelo
- Cada modelo resuelve una parte del problema usando distintos datos, algoritmos, atributos, o estimación de parámetros
- Tomo la decisión de manera conjunta (e.g. voto, promedio)
- Puedo mejorar la relación sesgo/varianza fácilmente, obteniendo modelos muy potentes
- Puedo usar scikit-learn o librerías específicas
- Es muy usado en competencias
- Es muy usado en la industria, por su capacidad en datos tabulados



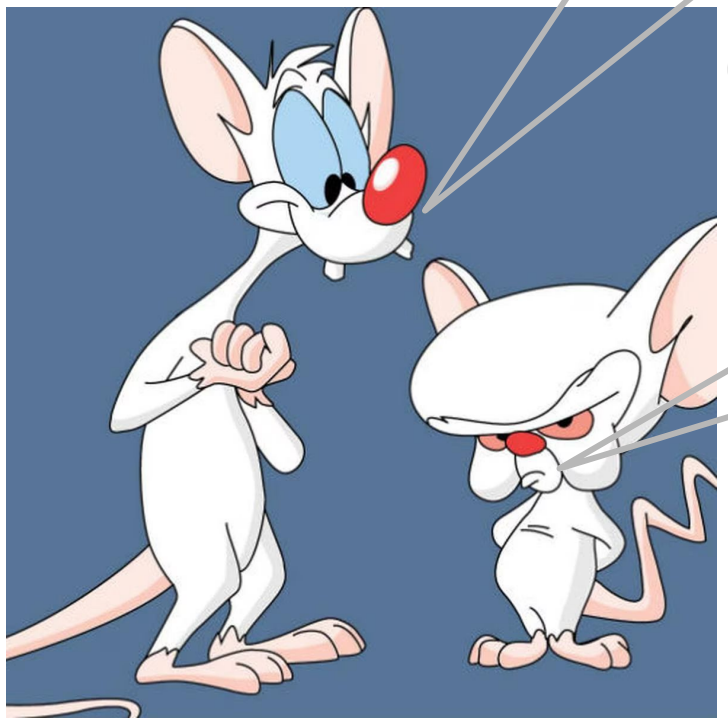
scikit learn

XGBoost

LightGBM

¿Qué pasa cuando los datos no son tabulados? (próxima unidad)

¿Qué vamos a hacer hoy?



Lo mismo que hacemos todas las noches… **vamos a los Colabs**