DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

# Aprendizaje Automático

**Clase 3:**

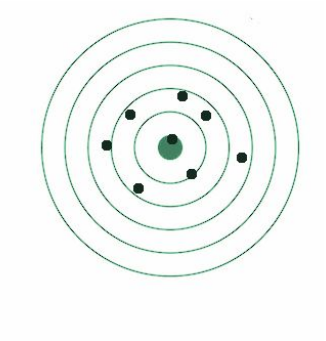
Teoría del error: Sesgo y Varianza
Evaluación y Selección de Modelos
Curvas de complejidad.

# ¿Qué debería saber después de esta clase?

- **Teoría el error** (sesgo y varianza).
- Diferencia entre **sobreestimación** y **sobreajuste**.
- Diferencia entre Model **Selection** y Model **Assessment**.
- Metodologías seguras de **separación de datos.** Entrenamiento, Desarrollo, Control.
- Técnicas **grid search**, **randomized search**, etc.
- Técnicas **cross validation**, **k-fold cross validation**, **group k-fold cross validation**.
- Entender las **curvas de complejidad.**

# Teoría del Error

## Sesgo y Varianza



"Las nociones de sesgo y varianza ayudan a explicar cómo los algoritmos muy simples pueden superar a los más sofisticados y cómo los ensambles pueden superar a los modelos individuales"

[Domingos, Pedro. "**A unified bias-variance decomposition**." Proceedings of 17th international conference on machine learning. Stanford: Morgan Kaufmann, 2000.]

http://homes.cs.washington.edu/~pedrod/bvd.pdf

# Tarea del aprendizaje supervisado

**Objetivo del aprendizaje supervisado**

Estimar la **función determinista f** que determina la relación $X \rightarrow Y$:

Es decir, estimar **f** tal que $\mathbf{Y = f(X)}$ a través de un modelo $\hat{h}_D$.

***Problema***: si la función **f** es determinista, ¿cómo puede ser que haya etiquetas contradictorias entonces?

*Ej:* $x^{(1)} = [1, 3, 14, 4] \rightarrow$ *etiquetado como Perro (es decir* $y^{(1)} = C_1$*)*
$x^{(2)} = [1, 3, 14, 4] \rightarrow$ *etiquetado como Gato (es decir* $y^{(2)} = C_2$*)*

***Posibles causas:***

- ***Variables no observadas:*** *El proceso* ***"real"*** *utiliza* ***más atributos***, ***no representados*** *en las dimensiones de X*
- ***Errores de medición:*** *Errores en la recolección de datos, como errores de redondeo, de registro, etc,*
- ***Errores de etiquetado:*** *Las etiquetas están mal, alguien (o algo) se* ***equivocó al etiquetar (o simplemente no hay acuerdo)***.
- ***Variación aleatoria en Y:*** *Puede haber una* ***variación aleatoria en la salida*** *que no está relacionada con la entrada.*
- ***Variación aleatoria en X:*** *En algunos casos, las variables de entrada pueden generarse mediante procesos estocásticos, como caminatas aleatorias, que crean una* ***aleatoriedad inherente o ruido.***

# Tarea del aprendizaje supervisado

**Objetivo del aprendizaje supervisado (revisado)**

Estimar la **función determinista f** que determina la relación $X \rightarrow Y$:

Es decir, estimar **f** tal que $\mathbf{Y = f(X) + \varepsilon}$ a través de un modelo $\hat{h}_D$. En donde $\varepsilon$ es el **"el error irreducible".**

- $\varepsilon$ **es una variable aleatoria** que representa la cantidad de ruido o incertidumbre en la relación entre las variables de entrada y la variable de salida. $\varepsilon$ podría depender de X, pero en general suponemos que no.

- Por ejemplo,
  - En regresión se espera que el término de error $\varepsilon$ siga una **distribución normal** con **media de cero** y **varianza finita**.
  - En clasificación podemos pensar algo que **cambia las respuestas al azar** (de 0 a 1 o 1 a 0, en caso binario) — por lo tanto sumar $\varepsilon$ es un abuso de notación.

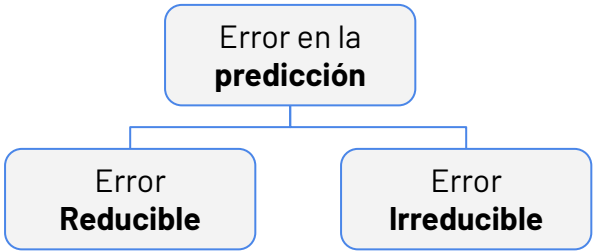
Esto es un "modelo"de la realidad (una serie de suposiciones que simplifica encarar el problema). Para más opciones o justificaciones, ver el ESLR (cap 2, sec 2.6)

# Representando el error de generalización

Hasta ahora vimos cómo minimizar el **error en entrenamiento** y **confiamos en que eso** ayudará para reducir el error de generalización.

Hoy estudiaremos **el error de generalización** y lo descompondremos para entender de dónde proviene.

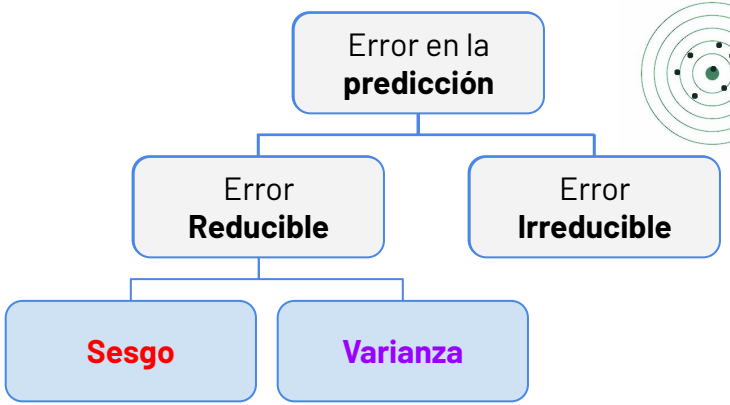
Nos concentraremos en la parte del error que **sí podemos reducir.**




Empecemos por caracterizar cuál será el error que esperamos encontrar al clasificar **una instancia** $\mathrm{x}^{(\mathrm{i})}$ al utilizar un modelo construido a partir del **algoritmo** $\mathrm{L}$

$$Error\_esperado(x^{(i)}; L) \ ?$$

# Sesgo y Varianza estadísticas




**Objetivo:** Aprender la relación $\mathbf{Y = f(X) + \varepsilon}$ a través de un modelo $\hat{h}_D(X)$.

$$Error\_esperado(x^{(i)}; L) = \mathrm{E}_{D_n}\left[error\left(y^{(i)}, \hat{h}_{(L,D_n)}(x^{(i)})\right)\right]$$

$$= \mathrm{E}_{D_n}\left[error\left(f(x^{(i)}) + \varepsilon, \hat{h}_{(L,D_n)}(x^{(i)})\right)\right]$$

$$\overset{\text{reg}}{=} \mathrm{E}_{D_n}\left[\left(f(x^{(i)}) + \varepsilon - \hat{h}_{(L,D_n)}(x^{(i)})\right)^2\right]$$

$$= \dots$$

$$= \left(\text{Sesgo}\left[\hat{h}_{(L,D_n)}(x^{(i)})\right]\right)^2 + \text{Var}\left[\hat{h}_{(L,D_n)}(x^{(i)})\right] + \text{Var}(\varepsilon)$$

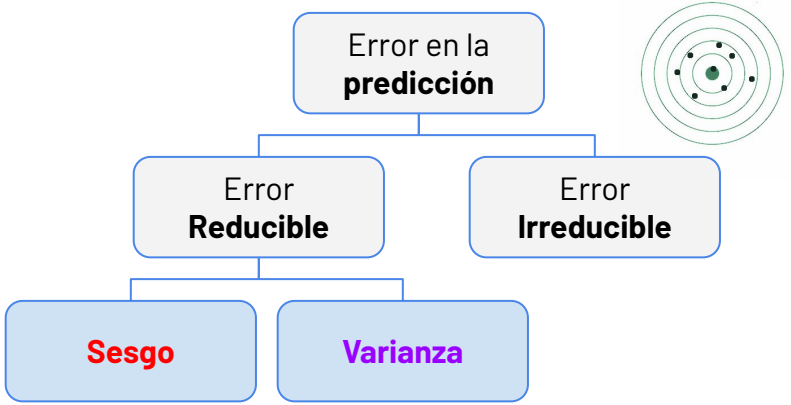
Para el caso de regresión.
Y tomando error = MSE.
MSE(a,b) = (b-a)$^2$

Ejercicio de la práctica "Bias-Variance decomposition"

En un ratito hablamos de clasificación.

En donde $\mathbf{E}_{\mathbf{Dn}}$ refiere a la esperanza sobre todos los posibles datasets (muestreados a partir de $\mathbf{P(X,Y)}$ de tamaño $n$)
$\hat{h}_{(L,Dn)}(x^{(i)})$ refiere a la predicción un modelo entrenado utilizando el algoritmo $L$ sobre los datos $D_n$)

# Sesgo y Varianza estadísticas




**Objetivo:** Aprender la relación $\mathbf{Y = f(X) + \varepsilon}$ a través de un modelo $\hat{h}_D(X)$.

$$Error\_esperado(x^{(i)}; L) \overset{\text{reg}}{=} \left( \text{Sesgo}\left[\hat{h}_{(L,D_n)}(x^{(i)})\right] \right)^2 + \text{Var}\left[\hat{h}_{(L,D_n)}(x^{(i)})\right] + \text{Var}(\varepsilon)$$

**Sesgo (bias): Dado un algoritmo,** cuánto esperamos que una predicción **difiera** del **valor real** (técnicamente, del valor medio real)

$$\text{Sesgo}\left[\hat{h}_{(L,D_n)}(x^{(i)})\right] = \text{E}_{D_n}\left[dif\left(\hat{h}_{(L,D_n)}(x^{(i)}), f(x^{(i)})\right)\right]$$

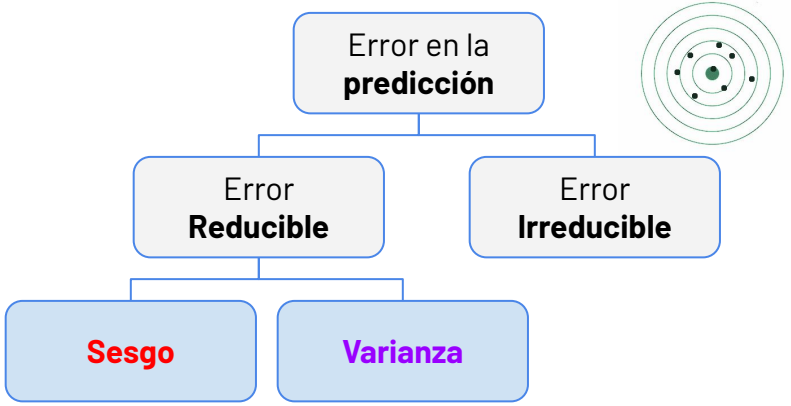
**Nota**, acá dif(a,b) = b-a (interesa el signo).

**Varianza: Dado un algoritmo,** cuánto esperamos que una predicción **difiera** del **valor más común de dicho algoritmo**.

$$\text{Var}\left[\hat{h}_{(L,D_n)}(x^{(i)})\right] = \text{E}_{D_n}\left[dif\left(\hat{h}_{(L,D_n)}(x^{(i)}), \text{E}_{D'_n}\left[\hat{h}_{(L,D'_n)}(x^{(i)})\right]\right)^2\right]$$

En donde $\mathbf{E_{Dn}}$ refiere a la esperanza sobre todos los posibles datasets (muestreados a partir de $\mathbf{P(X,Y)}$ de tamaño $n$)
$\hat{h}_{(L,Dn)}(x^{(i)})$ refiere a la predicción un modelo entrenado utilizando el algoritmo $L$ sobre los datos $D_n$)

# Sesgo y Varianza estadísticas




Simplificando un poco la notación

$$\begin{aligned} \mathrm{E} &\overset{\text{def}}{=} \mathrm{E}_{D_n} \\ pred^{(i)} &\overset{\text{def}}{=} \hat{h}_{(L,D_n)}(x^{(i)}) \\ \overline{pred}^{(i)} &\overset{\text{def}}{=} \mathrm{E}[pred^{(i)}] \end{aligned}$$

$$Error\_esperado(x^{(i)}; L) \overset{\text{reg}}{=} \left( \text{Sesgo}\left[\hat{h}_{(L,D_n)}(x^{(i)})\right] \right)^2 + \text{Var}\left[\hat{h}_{(L,D_n)}(x^{(i)})\right] + \text{Var}(\varepsilon)$$

**Sesgo (bias): Dado un algoritmo,** cuánto esperamos que una predicción **difiera** del **valor real** (técnicamente, del valor medio real)

$$\text{Sesgo}\left[pred^{(i)}\right] = \mathrm{E}\left[dif\left(pred^{(i)}, f(x^{(i)})\right)\right] = \mathrm{E}\left[f(x^{(i)}) - pred^{(i)}\right]$$

**Nota**, acá dif(a,b) = b-a (interesa el signo).

**Varianza: Dado un algoritmo,** cuánto esperamos que una predicción **difiera** del **valor más común de dicho algoritmo**.

$$\text{Var}\left[pred^{(i)}\right] = \mathrm{E}[dif\left(pred^{(i)}, \overline{pred}^{(i)}\right)^2] = \mathrm{E}[\left(pred^{(i)} - \overline{pred}^{(i)}\right)^2]$$

Nota:

Sesgo Estadístico != Sesgo inductivo
Sesgo Estadístico != Sesgo de Equidad (fairness)
Sesgo Estadístico != Bias term (redes / regresión)

# Nota sobre la $\mathbf{f}$ a estimar.

¿Por qué decimos **"valor real (técnicamente, del valor medio real)"**?

En la realidad seguramente tengamos **muchas instancias** con los **mismos atributos**, pero distintos resultados.

Ej, en predicción de valor de propiedades

- (100 mts$^2$, 3 baños) → $100.000
- (100 mts$^2$, 3 baños) → $130.000
- (100 mts$^2$, 3 baños) → $90.000

Esto es porque en realidad estamos muestrando de la distribución conjunta (X, Y) no de una función determinística (como ya discutimos).

Entonces, **¿cuál es "el valor real"** de una instancia del estilo (100 mts$^2$, 3 baños)? Habrá muchos valores reales posibles.

Tomamos a la media de esos valores como el valor real (por definición y porque parece algo razonable).

La $\mathbf{f()}$ desconocida**,** a estimar entonces es una función que creemos indica el valor medio real.

(más sobre este tema en la intro al paper)

# Nota: Sesgo promedio, Varianza promedio

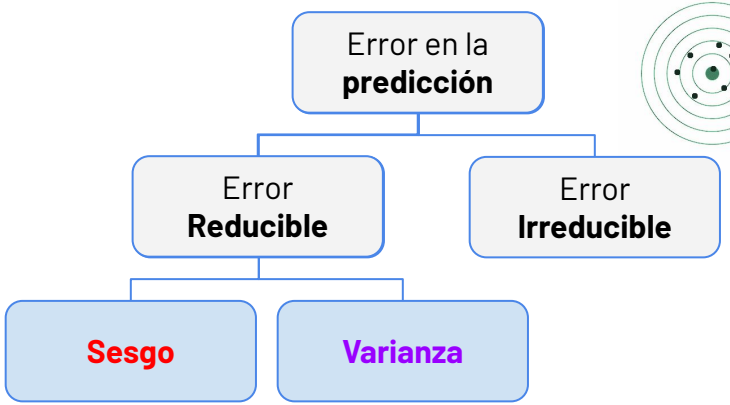
Es importante notar que esta definición sólo indica el sesgo y varianza de un algoritmo para **una instancia dada.**

Podemos definir el **sesgo y varianza del algoritmo** como la esperanza sobre las posibles instancias:

$$\text{Sesgo}(L) = \mathrm{E}_x \left[ \mathrm{E}_{D_n} \left[ \hat{h}_{(L,D_n)}(x) - f(x) \right] \right]$$

$$\text{Var}(L) = \mathrm{E}_x \left[ \mathrm{E}_{D_n} [ \left( \mathrm{E}_{D'_n} [\hat{h}_{(L,D'_n)}(x)] - \hat{h}_{(L,D_n)}(x) \right)^2 ] \right]$$

# Sesgo y Varianza estadísticas




**Objetivo:** Aprender la relación $\mathbf{Y = f(X) + \varepsilon}$ a través de un modelo $\hat{h}_D(X)$.

Y utilizando MSE como error

$$Error\_esperado(x^{(i)}; L) \overset{\text{reg}}{=} \left( \text{Sesgo} \left[ \hat{h}_{(L,D_n)}(x^{(i)}) \right] \right)^2 + \text{Var} \left[ \hat{h}_{(L,D_n)}(x^{(i)}) \right] + \text{Var}(\varepsilon)$$

$$Error\_esperado(x^{(i)}) \overset{\text{clf}}{=} (Sesgo \ldots Varianza \ldots \varepsilon)??$$

En clasificación, **pese a que la fórmula no sea la misma,** también puede expresarse la fórmula del error esperado en términos del sesgo y la varianza.

Domingos, Pedro. **"A unified bias-variance decomposition."** Proceedings of 17th international conference on machine learning. Stanford: Morgan Kaufmann, 2000 ].

(tienen que leer la intro para el cuestionario)

**Interesa entonces seguir entendiendo estos conceptos y ver cómo podemos manipularlos.**

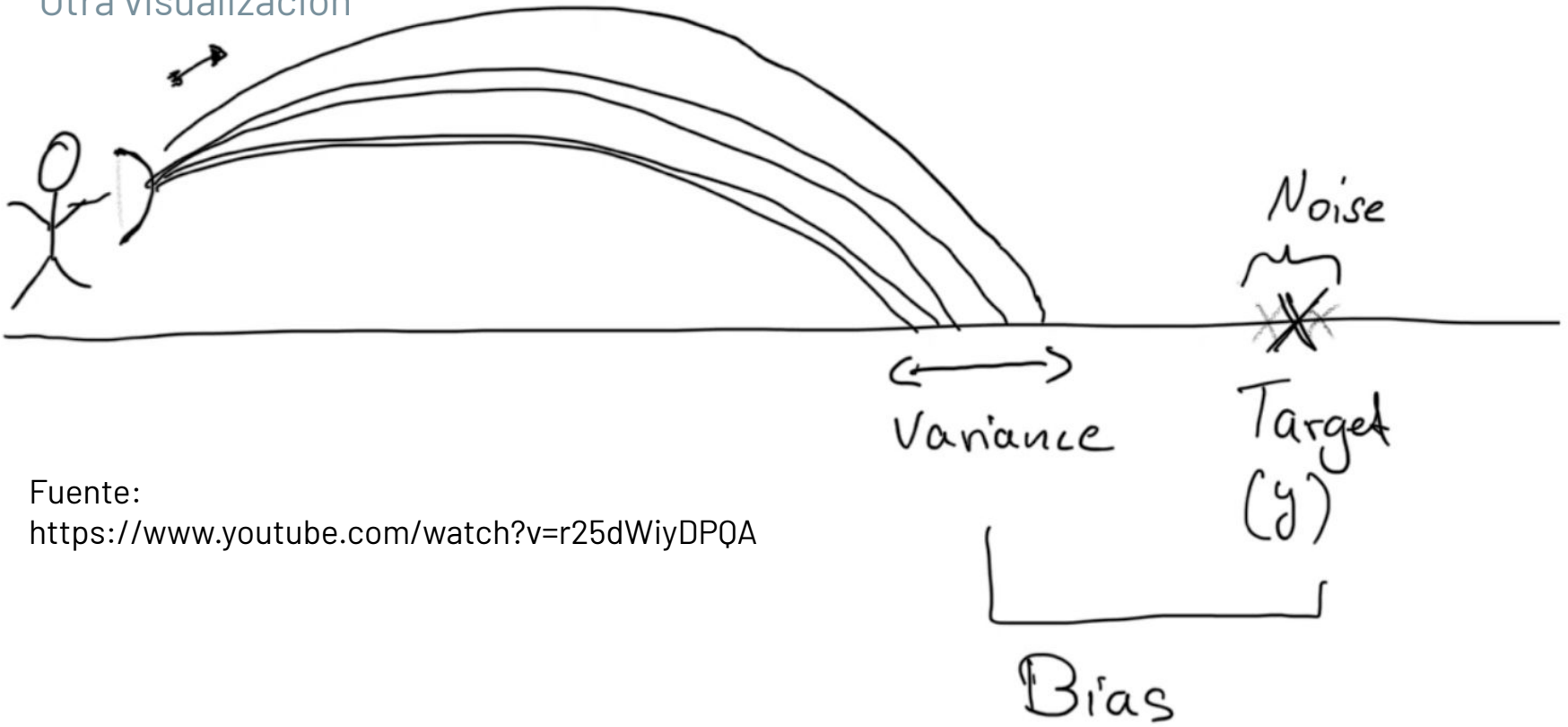
# Visualizaciones

$$\text{Sesgo}\left[pred^{(i)}\right] = \text{E}\left[dif\left(pred^{(i)}, f(x^{(i)})\right)\right]$$

$$\text{Var}\left[pred^{(i)}\right] = \text{E}[dif\left(pred^{(i)}, \overline{pred}^{(i)}\right)^2]$$

# Sesgo y Varianza

## Otra visualización



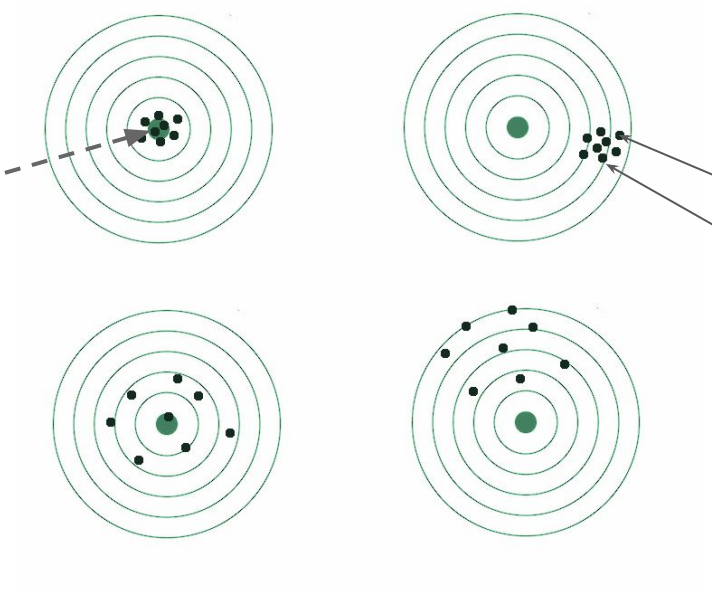

Fuente:
https://www.youtube.com/watch?v=r25dWiyDPQA

# Sesgo y Varianza

## Una visualización

$$\text{Sesgo}\left[pred^{(i)}\right] = \text{E}\left[dif\left(pred^{(i)}, f(x^{(i)})\right)\right]$$

$$\text{Var}\left[pred^{(i)}\right] = \text{E}[dif\left(pred^{(i)}, \overline{pred}^{(i)}\right)^2]$$



$f(x^{(i)})$

Valor “verdadero”de la instancia i (o valor verdadero medio si consideramos ε)

Predicciones para $x^{(i)}$ (al variar el dataset de entrenamiento para L)

$\hat{h}_{D1,\,L}(x^{(i)})$

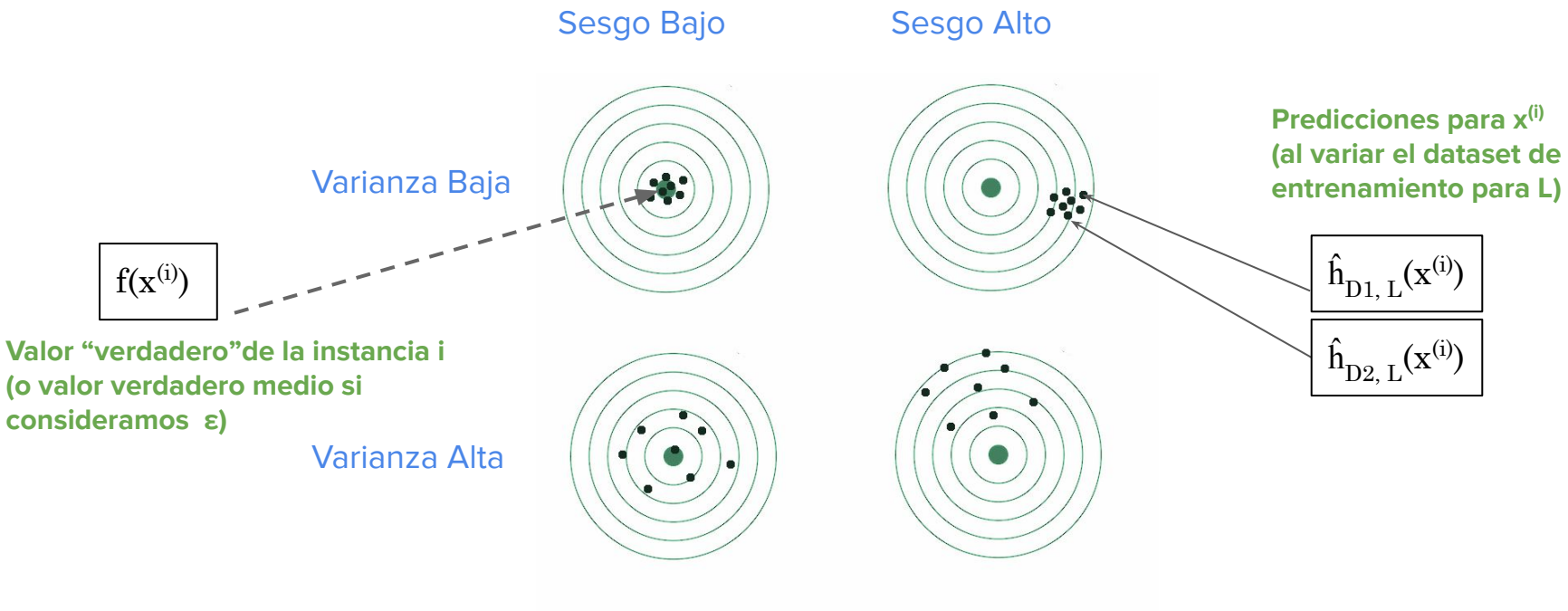
$\hat{h}_{D2,\,L}(x^{(i)})$

# Sesgo y Varianza

## Una visualización

$$\text{Sesgo}\left[pred^{(i)}\right] = \text{E}\left[dif\left(pred^{(i)}, f(x^{(i)})\right)\right]$$

$$\text{Var}\left[pred^{(i)}\right] = \text{E}[dif\left(pred^{(i)}, \overline{pred}^{(i)}\right)^2]$$

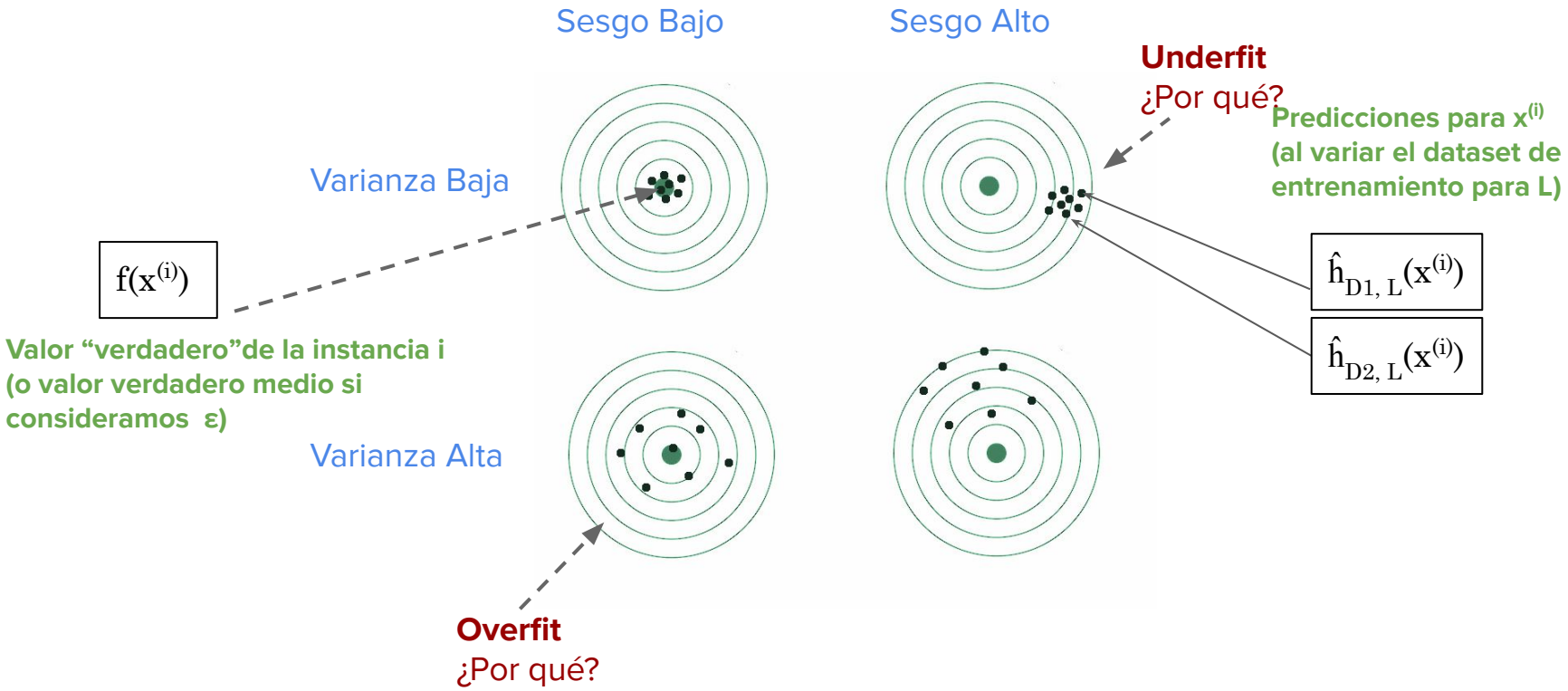


# Sesgo y Varianza

## Una visualización

$$\text{Sesgo}\left[pred^{(i)}\right] = \text{E}\left[dif\left(pred^{(i)}, f(x^{(i)})\right)\right]$$

$$\text{Var}\left[pred^{(i)}\right] = \text{E}[dif\left(pred^{(i)}, \overline{pred}^{(i)}\right)^2]$$

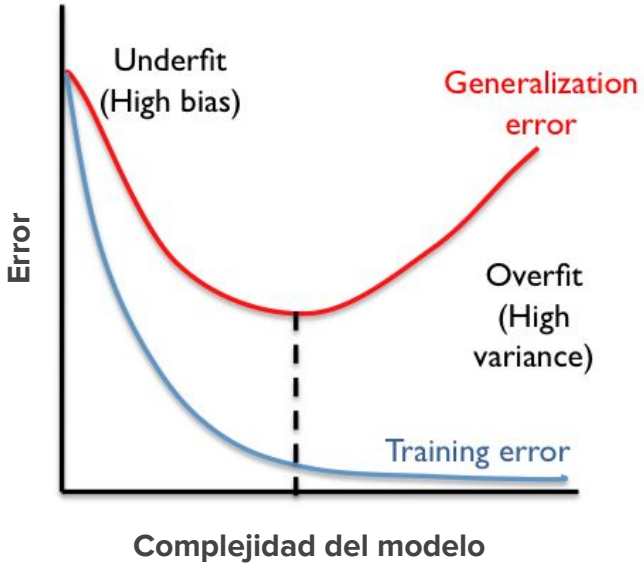


# Herramientas de Diagnóstico: Curvas de complejidad

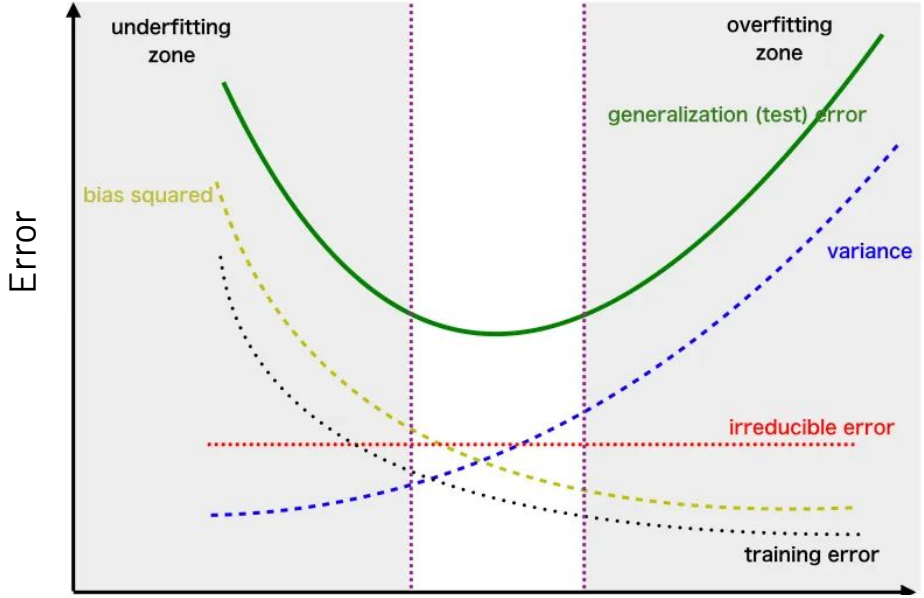
# Herramientas de diagnóstico

## Curvas de Complejidad del Modelo

**Procedimiento:**
Medir el error de entrenamiento y validación a medida que variamos hiperparámetros del algoritmo.

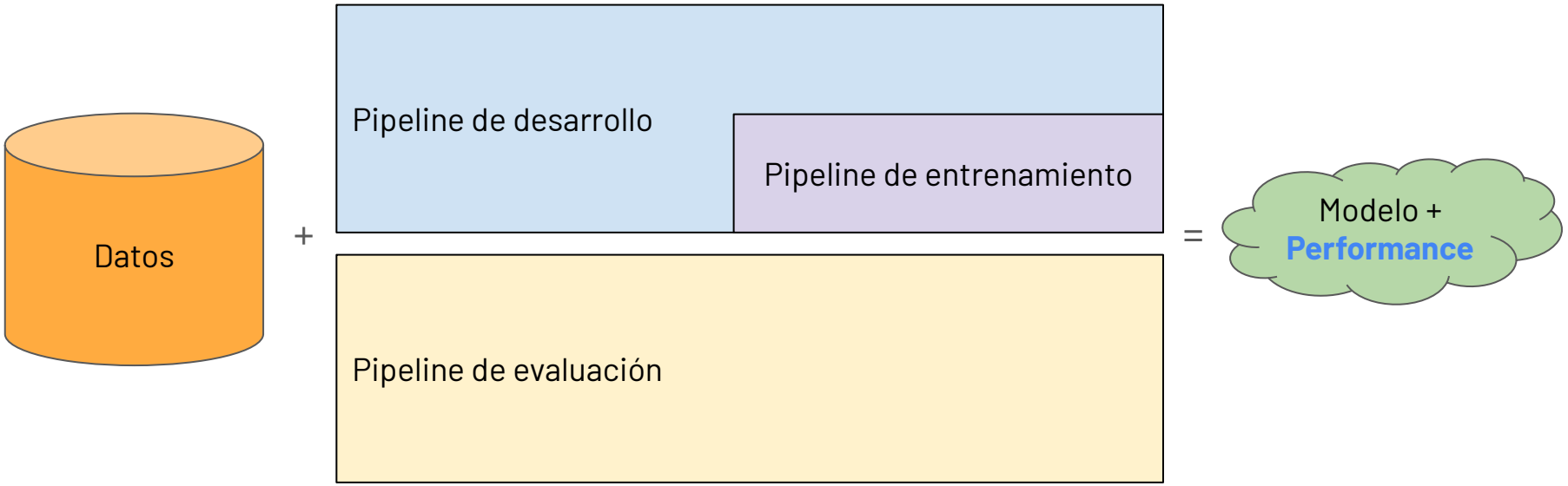



Cómo se descompone

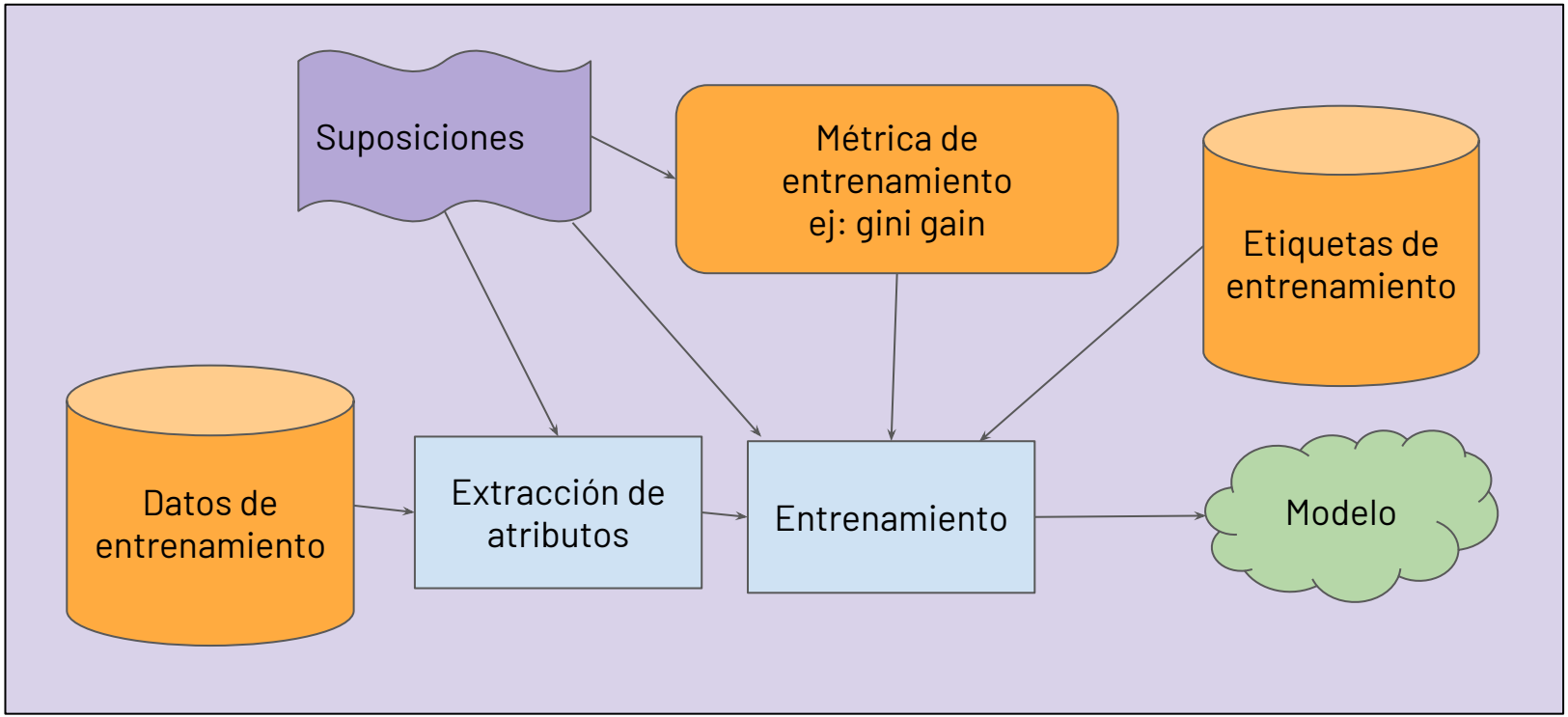


# Evaluación en la práctica

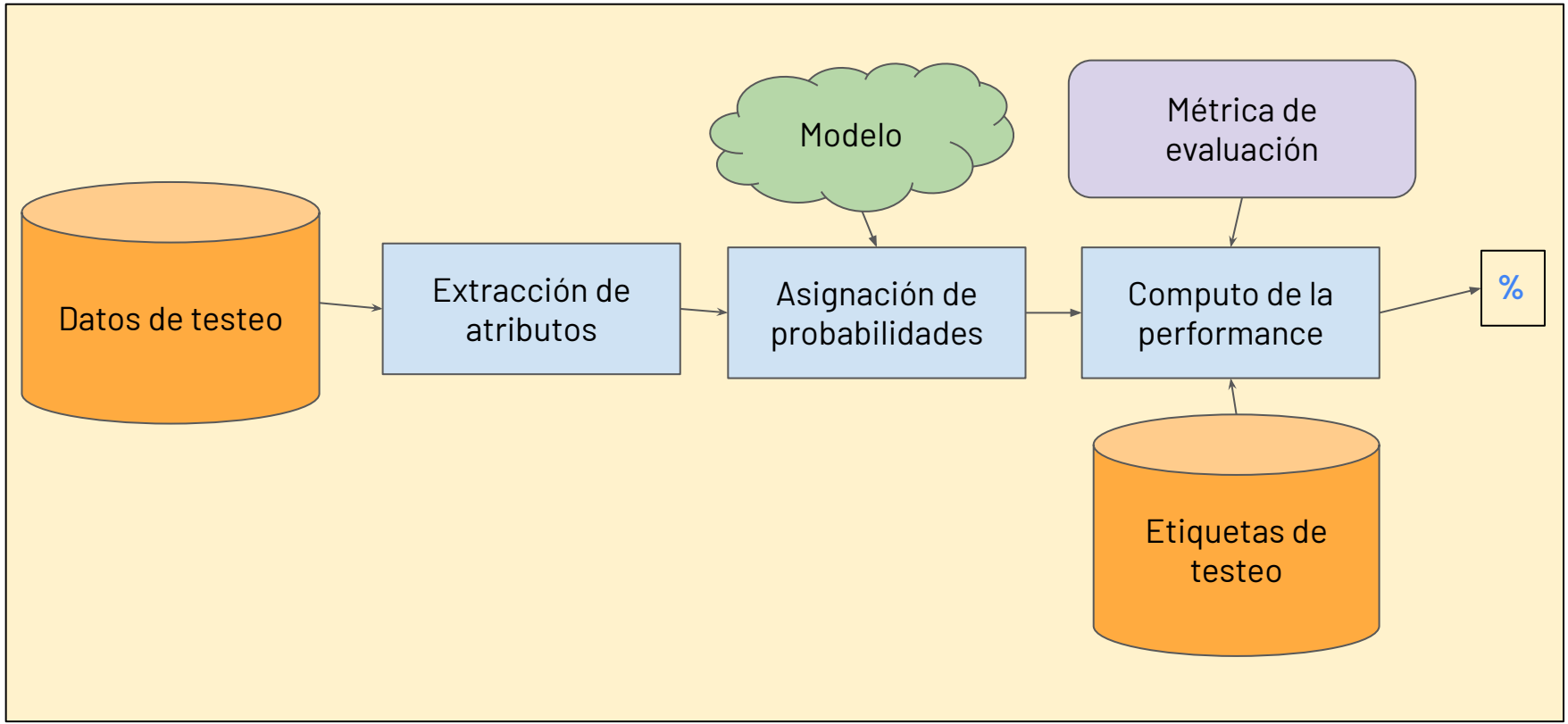
# Visión global



# Repaso pipeline de entrenamiento



# Pipeline de evaluación



# ¿Por qué enfocarse en la metodología de evaluación?

Casi todo lo que hacemos en Machine Learning depende en gran medida de la evaluación:

- ➢ **Comparamos** configuraciones de algoritmos mediante mediciones con alguna métrica.
- ➢ **Estimamos** la performance "en la realidad" (in the wild) y la reportamos a personas interesadas.

- ★ Es **fácil errar** en la creación de un modelo (entrenamiento), pero también es **fácil darse cuenta.**
- ★ Es **fácil errar** en la evaluación y **no es fácil darse cuenta**.
- ➢ No es raro ver un papers o sistemas en producción con problemas en la evaluación que no funciona.

Evaluar **bien** significa entender bien el **caso de uso,**

- entender qué me importa del problema y qué no,
- entender qué métrica/s refleja/n lo que quiero capturar,
- entender qué mecanismo de evaluación usar,
- entender **cómo no hacer/se trampa**
- entender **cómo hago que no me hagan trampa.**

# Selección y evaluación de modelos

Dos preguntas distintas:

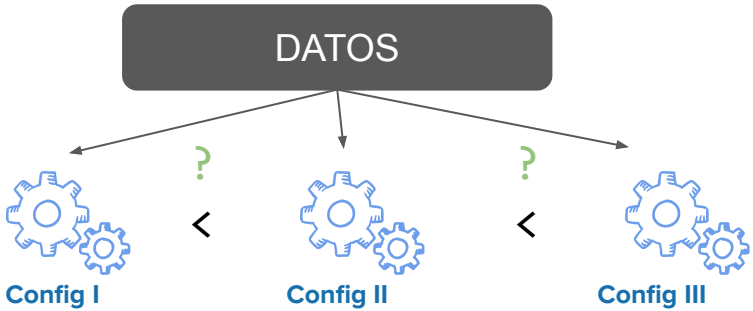
→ ¿Cómo **seleccionar** el mejor modelo entre varias opciones?

→ ¿Cómo **funcionará** mi modelo seleccionado en la "realidad"?

**Warning!**
Sobrecarga de la palabra "**modelo**". En estos casos se utiliza para referirse a la **configuración** y no la **hipótesis** resultante.

Podríamos llamar a las técnicas "Configuration Selection" / "Configuration Assessment".

**Model Selection:** Explorar distintas **configuraciones** de modelos (algoritmo + hiperparámetros + atributos) para poder seleccionar el mejor.




**Model Assessment (evaluación del modelo):** Una vez **seleccionada la mejor configuración,** estimar la performance que tendrá de la mejor manera posible.


¿Performance esperada en la naturaleza ("in the wild")?

# Selección de modelos

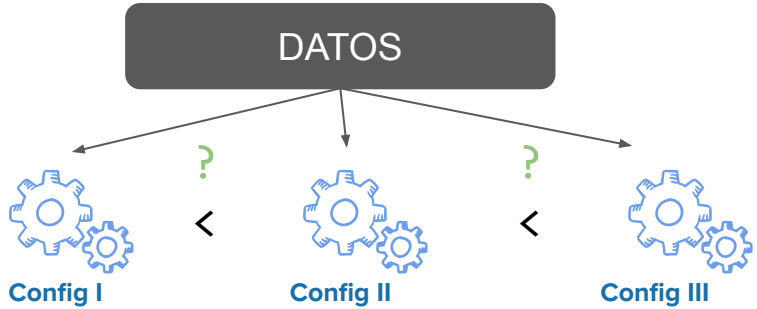
# Búsqueda del mejor modelo

**Terminología**

**Hiperparámetro:** Se refiere valores que se especifican manualmente en el algoritmo de aprendizaje antes de ejecutarlo.

**Parámetro:** Se refiere a valores internos del modelo resultante que se aprenden (ajustan) a partir de ejecutar el algoritmo sobre un dataset. Representan las reglas aprendidas.

———————— Por ejemplo en árboles ————————————
***Hiperparámetros**: Criterio de elección de atributos en cada nodo (Gini Gain, Information Gain), cantidad de hijos (árboles binarios vs. n-arios), criterio de parada (ej: max_depth), estrategia de poda, etc.*
**Parámetros**: *las triplas <nood_id, atributo, corte> obtenidos. ("Lo que se guarda si hago un **modelo.zip**" )*




**¿Cómo buscar la mejor configuración?**

Para poder elegir un modelo final, es natural explorar distintas combinaciones de **hiperparámetros**, distintos **algoritmos** de aprendizaje y distintas **formas de extraer o procesar atributos.**

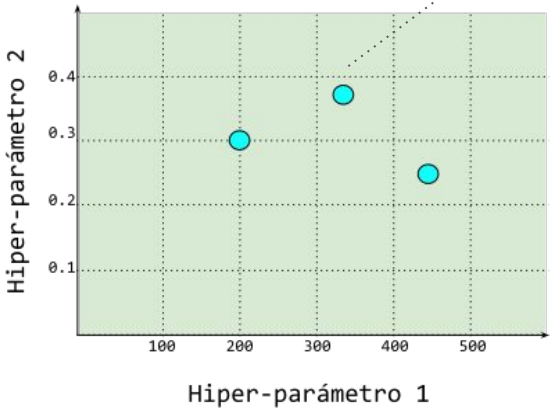
# Búsqueda del mejor modelo

## Métodos de exploración de combinaciones: Grid, Random

**Conf 83 ($hiper_1$=350, $hiper_2$=0.4)**
Accuracy: 0.83

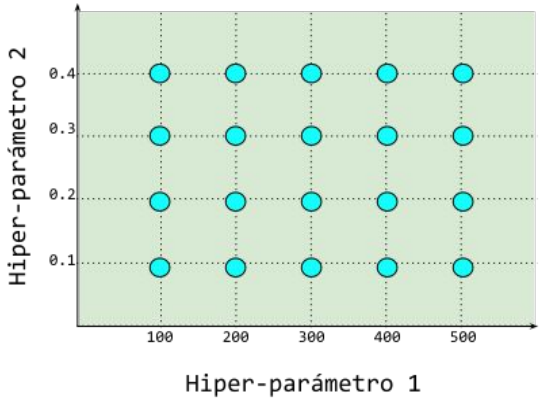
### Manual Search

Setear a mano hiper-parámetros que pensamos que van a funcionar bien.




### Grid Search

Plantear opciones y explorar todas las combinaciones.




**Ejemplo:**

- **5** hiperparámetros.
- **10** valores diferentes para cada híper.
- **100,000** ($10^5$) evaluaciones.
- Entrenamiento del modelo, promedio de **10 minutos**
- La optimización de hiperparámetros nos llevaría casi **2 años.**

Bergstra, James, and Yoshua Bengio. "Random search for hyper-parameter optimization." *Journal of Machine Learning Research* 13.Feb (**2012**)

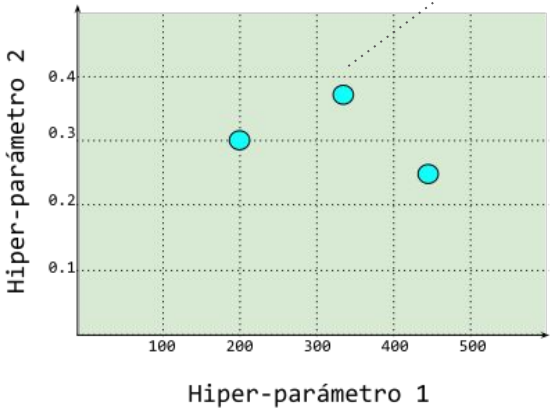
# Búsqueda del mejor modelo

## Métodos de exploración de combinaciones: Grid, Random

**Conf 83 ($hiper_1$=350, $hiper_2$=0.4)**
Accuracy: 0.83

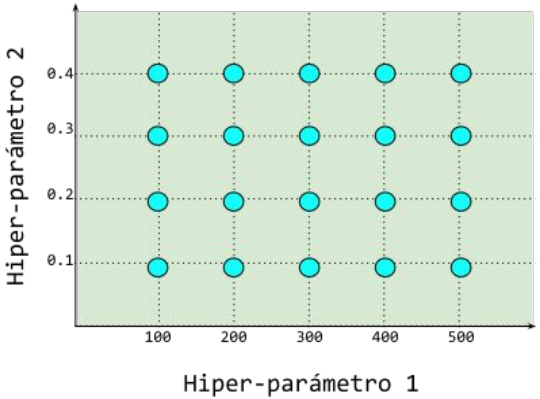
### Manual Search

Setear a mano hiper-parámetros que pensamos que van a funcionar bien.



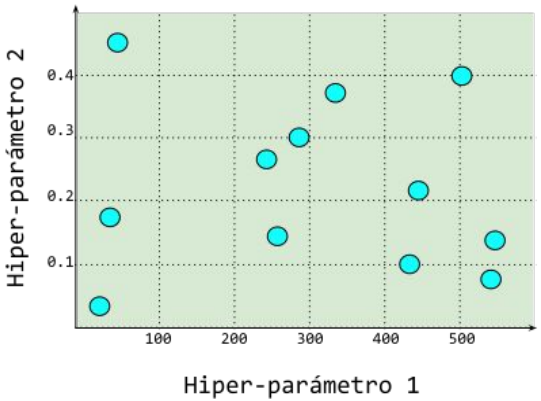

### Grid Search

Plantear opciones y explorar todas las combinaciones.




### Random Search

Se plantean distintas opciones y se exploran combinaciones al azar.




Bergstra, James, and Yoshua Bengio. "Random search for hyper-parameter optimization." *Journal of Machine Learning Research* 13.Feb (**2012**)

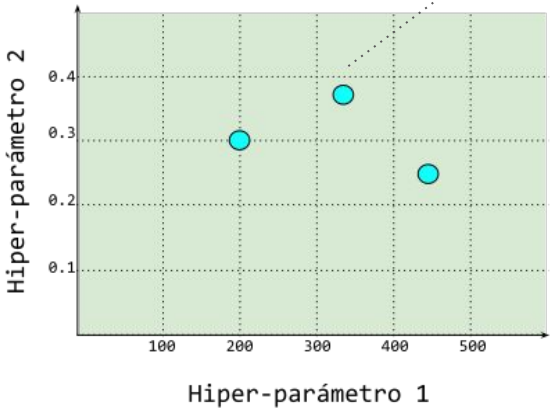
# Búsqueda del mejor modelo

## Métodos de exploración de combinaciones: Grid, Random

**Conf 83 ($hiper_1$=350, $hiper_2$=0.4)**
Accuracy: 0.83

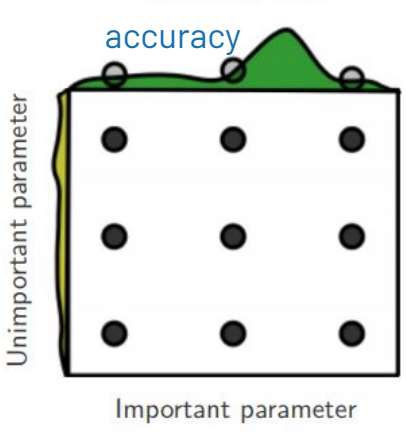
### Manual Search

Setear a mano hiper-parámetros que pensamos que van a funcionar bien.



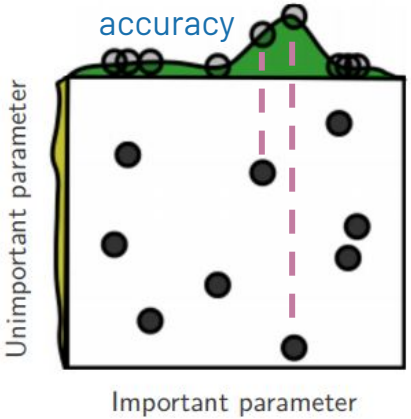

### Grid Search

Plantear opciones y explorar todas las combinaciones.




### Random Search

Se plantean distintas opciones y se exploran combinaciones al azar.




Bergstra, James, and Yoshua Bengio. "Random search for hyper-parameter optimization." *Journal of Machine Learning Research* 13.Feb (**2012**)

# Búsqueda del mejor modelo

## Métodos de exploración de combinaciones: Random

**Especificando distribuciones**

En random search, se puede especificar de qué manera muestrear al azar los hiperparámetros.

| Parameter name | Distribution | Values |
|---|---|---|
| Step size | Log-uniform | $x \in [0.01, 0.5]$ |
| Batch size | Log-uniform integer | $x \in [16, 512]$ |
| Activation function | Categorical | $x \in \{\text{Relu}, \text{PRelu}, \text{Elu}, \text{Sigmoid}, \text{Tanh}\}$ |
| Number of hidden layers | Categorical | $x \in \{1, 2\}$ |
| Number of units in the first layer | Uniform integer | $x \in [50, 1000]$ |
| Number of units in the second layer (In case if it defined) | Uniform integer | $x \in [50, 1000]$ |
| Dropout layer | Uniform | $x \in [0, 0.5]$ |

Ejemplo scikit-learn (SVM)

```
{'C': scipy.stats.expon(scale=100), 'kernel': ['rbf'], 'class_weight':['balanced', None]}
```

(ver https://scikit-learn.org/stable/modules/grid_search.html#grid-search)

# Búsqueda del mejor modelo

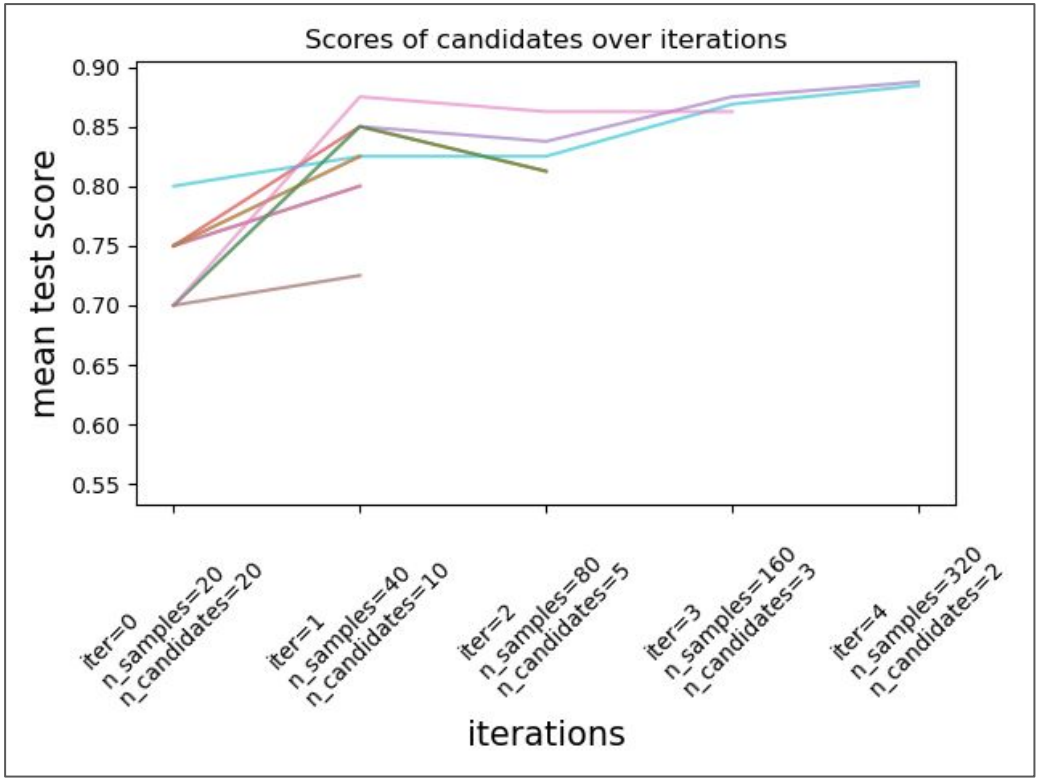
## Métodos de exploración de combinaciones: Halving ("reducir a la mitad")

Una alternativa a **grid/randomized search** es:

1) Darle pocos "recursos" a las primeras iteraciones (entrenar muchos modelos con pocos datos, con pocas iteraciones)
2) Seleccionar los más prometedores
3) Volver a probar con cada vez más recursos.

- **"Halving" Random Search**
- **"Halving" Grid Search**
- **Hyperband**

Ver:
https://scikit-learn.org/stable/auto_examples/model_selection/plot_successive_halving_heatmap.html#sphx-glr-auto-examples-model-selection-plot-successive-halving-heatmap-py



# Búsqueda del mejor modelo

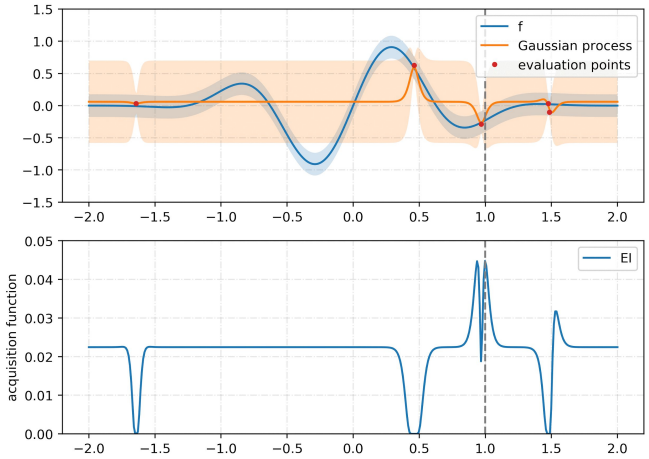
## Métodos de exploración de combinaciones: Optimización Bayesiana

Los métodos anteriores **no toman en cuenta** la información provista por **evaluaciones anteriores** para decidir cuál combinación probar a continuación.

Nos interesaría construir un modelo $P(s \mid h)$ en donde $\mathbf{s}$ es un puntaje de error (mientras más chico mejor), y $\mathbf{h}$ un valor para el hiperparámetro. Estimar $P(s \mid h)$, basado en observaciones anteriores, permite buscar en áreas prometedoras.

Métodos de esta pinta (aka **"Secuencial Model Based Optimization"**)

- Gaussian Process with Expected Improvement
- Random Forests (lo veremos más adelante)
- **Tree-structured Parzen Estimators (TPE)**
  (no lo veremos en esta edición, quedan las slides al final de la clase)

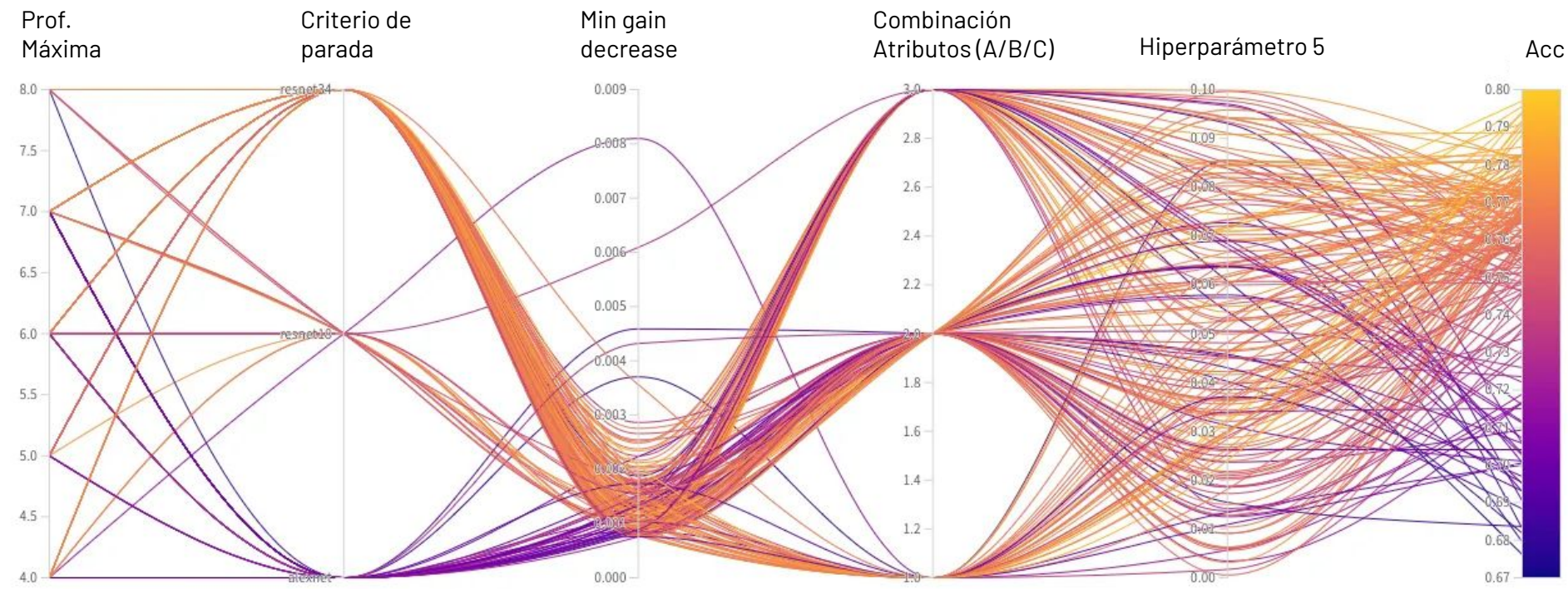



Fuente:
https://techblog.criteo.com/hyper-parameter-optimization-algorithms-2fe447525903

[Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. (2011). "**Algorithms for hyper-parameter optimization**"]
[Bergstra, J., Yamins, D., & Cox, D. (2013, February). "**Making a science of model search: Hyperparameter optimization in hundreds of dimensions…**"]

# Búsqueda del mejor modelo

## Ejemplo de visualización de resultados




**Fuente**: https://wandb.ai/stacey/pytorch_intro/reports/Meaning-and-Noise-in-Hyperparameter-Search-with-Weights-Biases--VmlIdzo0Mzk5MQ

# Evaluación de una configuración dada

La expresión "**test**" es ambigua.
En esta materia evitaremos usarla.

# Estimación de performance
## Validación Cruzada

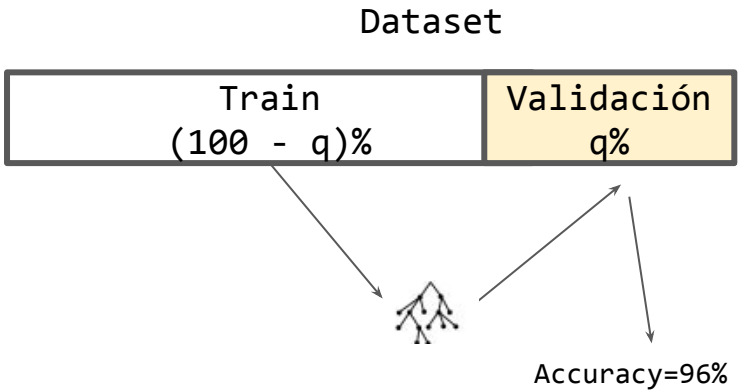
¿Cómo estimar la performance de una configuración?

Medir accuracy sobre datos de entrenamiento → mala idea. ¿Por qué?

Surge la necesidad de separar un **q%** de datos, para validar los modelos: datos de validación (o test).

Los datos se deben separar al azar(*) para evitar cualquier orden o estructura subyacente en los datos.

### Validación Cruzada (cross validation)(**)




(*) **Esto no siempre es así.** Veremos en un par de slides

(**) Algunos autores llaman "Cross Validation" a lo que veremos en la diapo siguiente y esto lo nombran distinto (train/test split) por ejemplo

# Estimación de performance

## K-Fold Cross Validation

- ¿Qué puede pasar si tenemos mala suerte al separar los datos para entrenamiento/validación? Estamos midiendo sólo sobre el **q%** (en general 5%, 10% o 20%) de nuestros datos. ¿El resultado será confiable? La estimación de performance del modelo podría no ser realista.

- Estos problemas surgen **especialmente** cuando tenemos **pocos datos**, si no, la validación cruzada suele alcanzar.

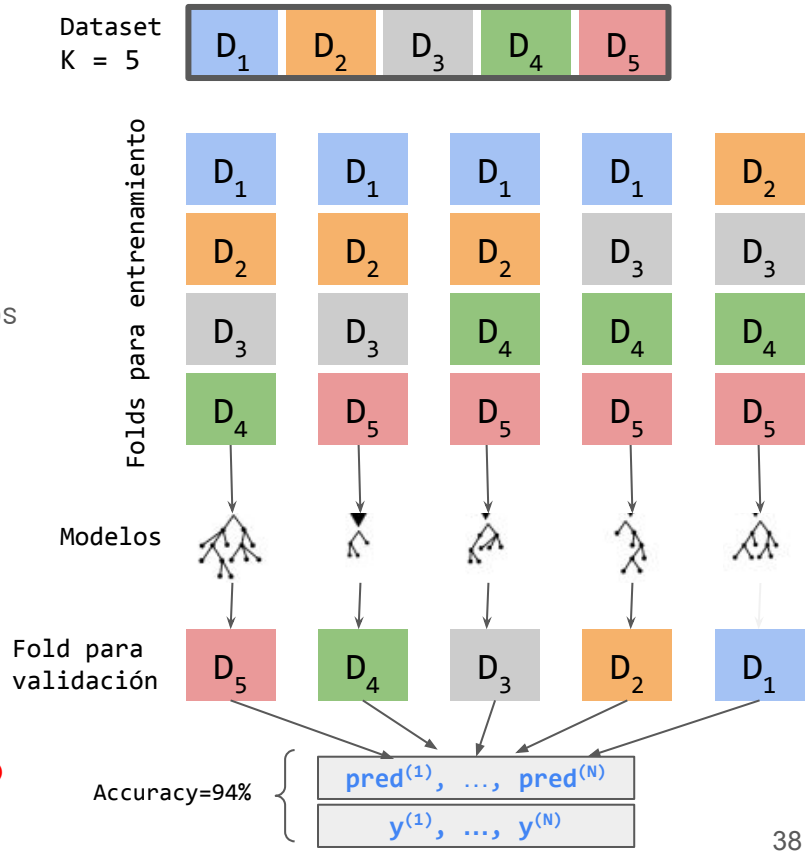
- Con pocos datos, sería más útil poder probar nuestro algoritmo de aprendizaje con todos nuestros datos. Surge la idea de Validación Cruzada de "k" iteraciones: **K-Fold Cross Validation**

# Estimación de performance

## K-Fold Cross Validation

Dado **L** (un algoritmo de aprendizaje con ciertos hiperparámetros ya establecidos) y el dataset **D:**

1. Separamos **D** en **K subconjuntos** a los que llamamos **folds**: $D_1, D_2, D_3, \ldots, D_K$.
2. Construimos **K modelos** que serán entrenados utilizando todos los datos (salvo los del k-ésimo fold). Es decir, $f_{L,D-Dk} = L(D-D_k)$.
3. Para cada $x^{(i)} \in D$:
   - $pred^{(i)} = f_{L,D-Dk}(x^{(i)})$
     `# predecimos utilizando el modelo que no vio ese dato en entrenamiento.`
   - `predicciones[i] = pred`$^{(i)}$
     `# juntamos las predicciones como si vinieran todas del mismo modelo`
4. Computamos alguna métrica del error sobre el conjunto entero[(*)] **predicciones** vs **y**






(*) Algunos autores sugieren un abordaje distinto, en donde se calcula la métrica para cada fold de validación por separado y después se promedia el resultado (no lo

# Estimación de performance

## K-Fold Cross Validation

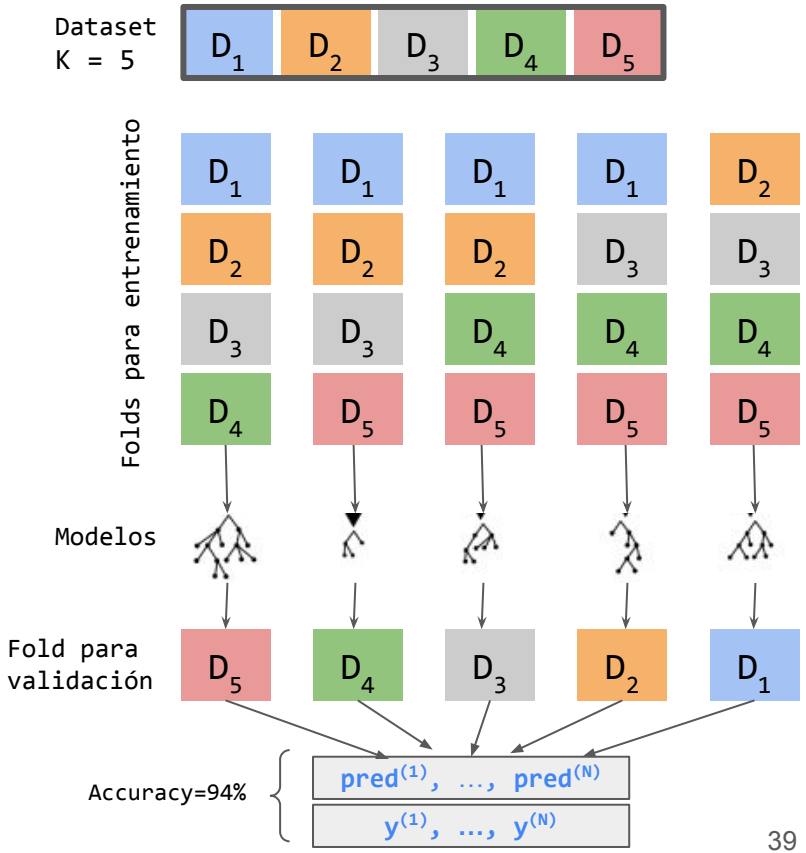
¿Qué sucede si **K = N**? (esto se llama **leave-one-out cross validation**)

- ¿Costo computacional?
- ¿Qué tan similares / distintos son los K modelos en ese caso?

Perfecto, tengo **K** modelos. **¿Cuál uso de ahora en más?**

**Opciones:**

- Hacer una votación ("Ensamble" de modelos)
- Usar el "mejor" (viendo los resultados por fold, pero con el riesgo de haber elegido un fold de validación "más fácil").
- ¿Re-entrenar utilizando todos los datos? OK, pero veremos **algunos riesgos** en breve.



# Estimación de performance

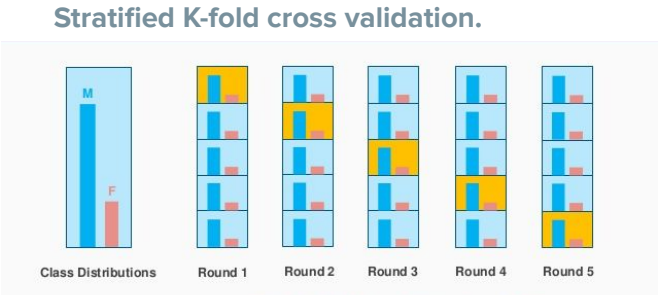
## Variaciones de K-Fold Cross Validation

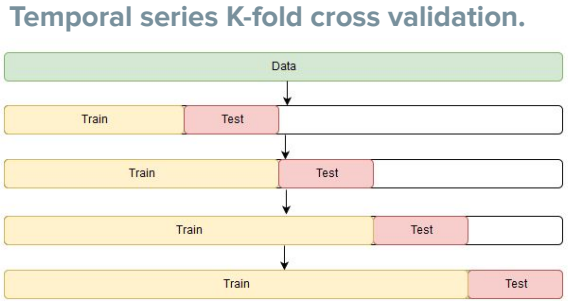
¿Es buena idea hacer **K folds al azar**?

- ¿Las clases están desbalanceadas?
- ¿Orden temporal de los datos?
- ¿Orden espacial? (datos regionales)

Pensamiento clave **¿Qué queremos aprender?**

Preguntarse entonces: ¿Tengo la misma cantidad de información que tendré en test (o en la realidad)? ¿Quiero mejorar mi predicción a costa de incluir esa información? (ej: máquina de imagenes médica)






Estas técnicas y más en Müller, A. C., & Guido, S. (2016). Introduction to machine learning with Python: a guide for data scientists.

[YE, Wenqian, et al. **Spurious correlations in machine learning:** A survey. 2024. https://arxiv.org/pdf/2402.12715v1]

# Group K-fold

¿Qué sucede con instancias no independientes? (casi siempre)

Clave, separar por "condición" (todo lo que que introduzca correlaciones espurias o que faciliten la tarea).

Ejemplos (para pensar ahora)

- **$Ej_1$: Detección de emociones**, tenemos 2000 instancias, pero de 10 hablantes. ¿Cómo hacemos las particiones en folds? ¿Cuál es la pregunta? ¿A qué quiero generalizar?

- **$Ej_2$: Detección de covid en toces:** Micrófono utilizado. ¿puede un modelo aprender a detectar el micrófono?

Group-k-fold cross validation. Asegúrese de que los datos de diferentes grupos no se mezclen entre los conjuntos de entrenamiento y prueba.

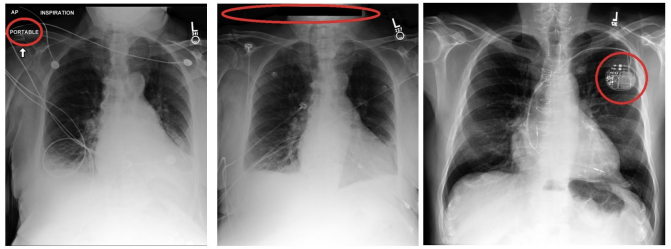
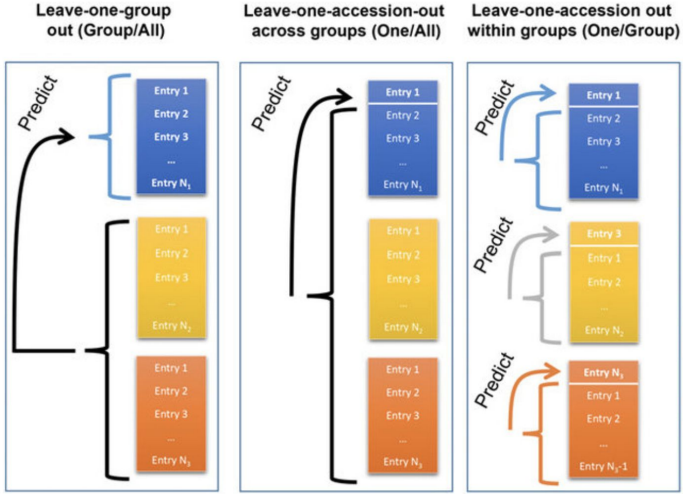


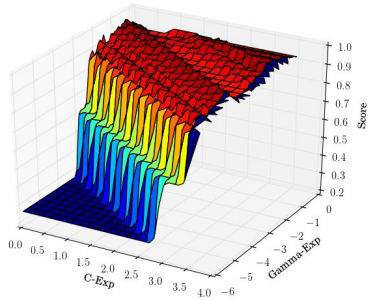

Figure 2: Hospital tags, strips, and medical devices exemplify several unknown group labels in the MIMIC-CXR dataset, which might spuriously correlate with the ground truth diagnosis results.




Fuente: [Prospects of genomic prediction in the USDA Soybean Germplasm Collection: Historical data creates robust models for enhancing selection of accessions]

# Volviendo a la selección del mejor modelo

Una vez probadas distintas combinaciones de estructura del modelo e hiperparámetros obtenemos un **puntaje por cada combinación**




(Utilizando cross validation)

Una vez seleccionados los mejores hiperparámetros, **podemos entrenar un modelo utilizando todos nuestros datos.**

**Escenario frecuente:**

Construimos nuestro modelo con la combinación de **atributos + algoritmos + hiperparámetros** con mejor desempeño.

Lo ponemos a funcionar con datos nuevos, y los resultados son **peores**.

¿Qué falló?

A otro nivel, repetimos el mismo error de antes. Evaluamos un modelo sobre los **mismos datos** que usamos para **construirlo (esta vez no "para entrenarlo")**.

- Entonces, sobreestimamos la performance de nuestro modelo.
- ¿Solución?

# ¿Por qué daría mal?

Ya vimos que nuestros modelos pueden sobreajustar en el conjunto de entrenamiento.

**Sobreajuste en el conjunto de prueba:** Otra noción de sobreajuste es la brecha entre el rendimiento en el conjunto de prueba y el rendimiento en la distribución subyacente de los datos.

Al adaptar las elecciones de diseño del modelo al conjunto de prueba, la preocupación es que **ajustemos implícitamente los hiperparámetros** al conjunto de prueba. La precisión en el conjunto de prueba, entonces, pierde su validez como una medida precisa del rendimiento en datos verdaderamente no vistos.

**Opinión personal:**
Una de las causas es:
*regression to the mean*
*(regression to the mediocrity)*
https://www.youtube.com/watch?v=1tSqSMOyNFE
(Veritasium)

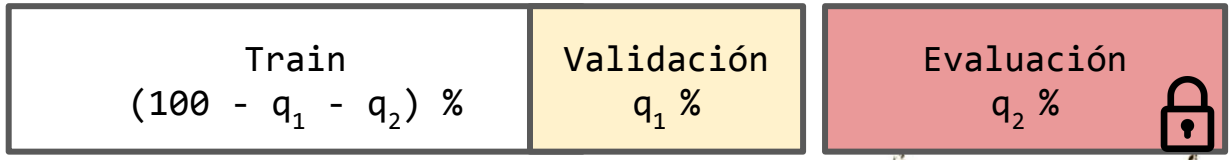
# Evaluación del modelo final

# Desarrollo - Evaluación

- Lo antes posible separamos un **conjunto para evaluación.**
- Serán datos que no se toquen **hasta el final**.
- Todas las pruebas y ajustes se hacen sobre el conjunto para **Desarrollo**.
- Cuando **termina el desarrollo**, se **evalúa** sobre los datos de evaluación separados.
- La estimación de performance será la más realista.
- Ese valor es el que se reporta. **¡No volver atrás!**

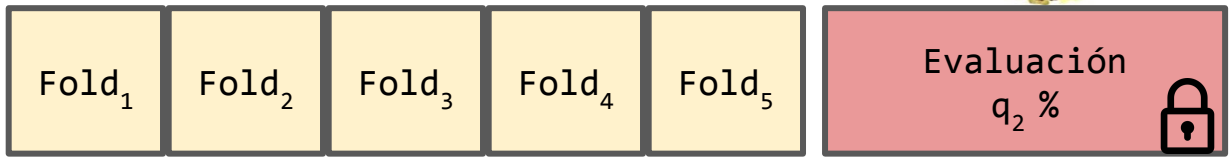
También llamado "**test set**", "**eval set**", "**hold out set**", "**held-out set**", "**control set**" depende la bibliografía.

Caso general




Desarrollo


Caso pocos datos
(K-fold cross val)




Desarrollo

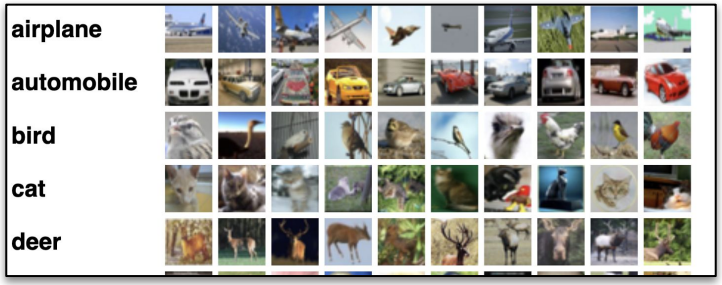
# Desarrollo - Evaluación

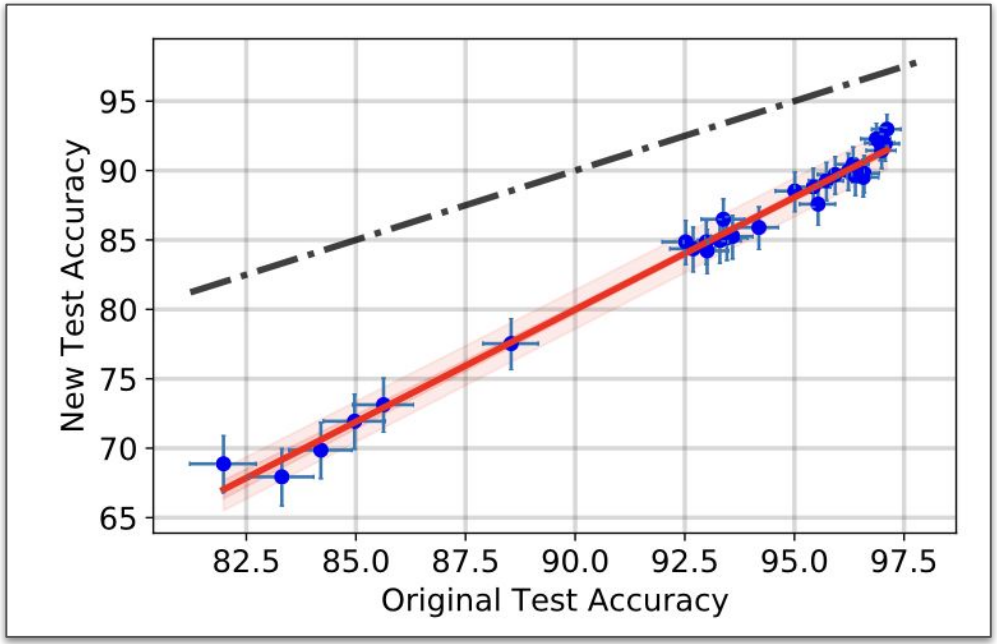
¡Importante: Los eval sets deberían ser actualizados constantemente!

Ejemplo: https://arxiv.org/abs/1806.00451
Recrearon un dataset equivalente al original y notaron que los modelos bajaron su performance de 4% to 10%

Do CIFAR-10 Classifiers Generalize to CIFAR-10?

Benjamin Recht, UC Berkeley
Rebecca Roelofs, UC Berkeley
Ludwig Schmidt, MIT
Vaishaal Shankar, UC Berkeley




| | Original Accuracy | New Accuracy | Gap | Δ Rank |
|---|---|---|---|---|
| shake_shake_64d_cutout [3, 4] | 97.1 [96.8, 97.4] | 93.0 [91.8, 94.0] | 4.1 | 0 |
| shake_shake_96d [4] | 97.1 [96.7, 97.4] | 91.9 [90.7, 93.1] | 5.1 | -2 |
| shake_shake_64d [4] | 97.0 [96.6, 97.3] | 91.4 [90.1, 92.6] | 5.6 | -2 |
| wide_resnet_28_10_cutout [3, 22] | 97.0 [96.6, 97.3] | 92.0 [90.7, 93.1] | 5 | +1 |
| shake_drop [21] | 96.9 [96.5, 97.2] | 92.3 [91.0, 93.4] | 4.6 | +3 |
| shake_shake_32d [4] | 96.6 [96.2, 96.9] | 89.8 [88.4, 91.1] | 6.8 | -2 |
| darc [11] | 96.6 [96.2, 96.9] | 89.5 [88.1, 90.8] | 7.1 | -4 |
| resnext_29_4x64d [20] | 96.4 [96.0, 96.7] | 89.6 [88.2, 90.9] | 6.8 | -2 |

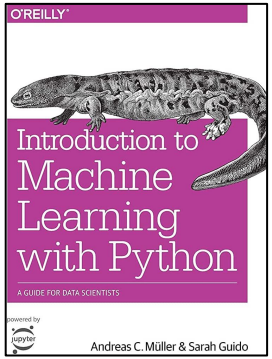
# TAREA

- Leer el **capítulo 5** "**Model Evaluation and Improvement**" (hasta Evaluation Metrics and Scoring, sin incluir) de Müller, A. C., & Guido, S. (2016). Introduction to machine learning with Python: a guide for data scientists.

- Leer (al menos) **la introducción del paper y las definiciones de la sección 2 del paper:** Domingos, Pedro A unified bias-variance decomposition (y tanto como puedan de los teoremas que se plantean)

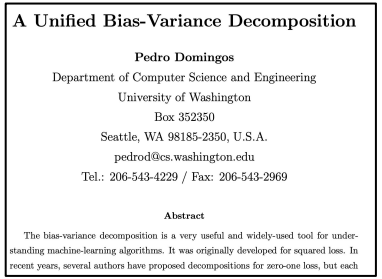
- Completar el cuestionario

(Opcional) Raschka, S. (2018). **Model evaluation, model selection, and algorithm selection in machine learning.** arXiv preprint arXiv:1811.12808. https://arxiv.org/abs/1811.12808




Andreas C. Müller

Sarah Guido



A Unified Bias-Variance Decomposition

Pedro Domingos
Department of Computer Science and Engineering
University of Washington
Box 352350
Seattle, WA 98185-2350, U.S.A.
pedrod@cs.washington.edu
Tel.: 206-543-4229 / Fax: 206-543-2969

Abstract

The bias-variance decomposition is a very useful and widely-used tool for understanding machine-learning algorithms. It was originally developed for squared loss. In recent years, several authors have proposed decompositions for zero-one loss, but each

Pedro Domingos

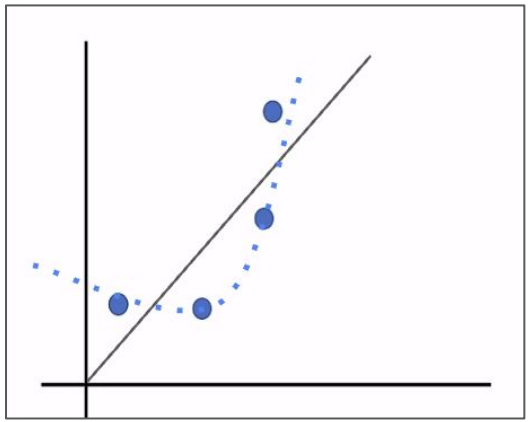
Sebastian Raschka

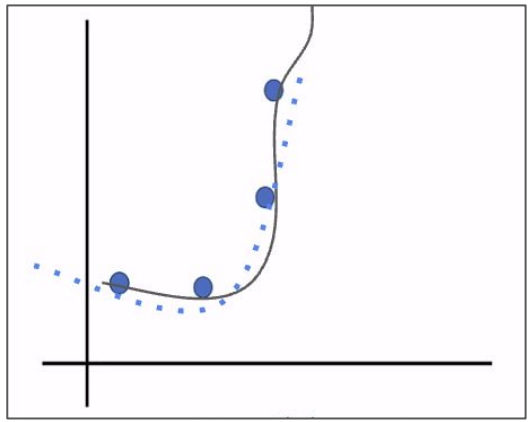
# Otra visualización para sesgo y varianza (en regresión)

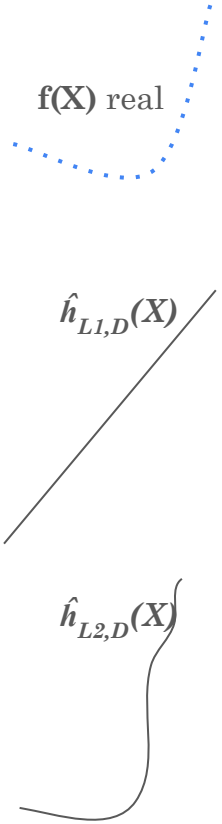
# Sesgo varianza visualización (regresión)

Usaremos para este ejemplo la **visualización "convencional" (no la venimos usando en clase)** para modelos de regresión (eje x = atributos, eje y = predicciones)

Acá vemos modelos distintos ajustados a los mismos datos







# Sesgo y Varianza

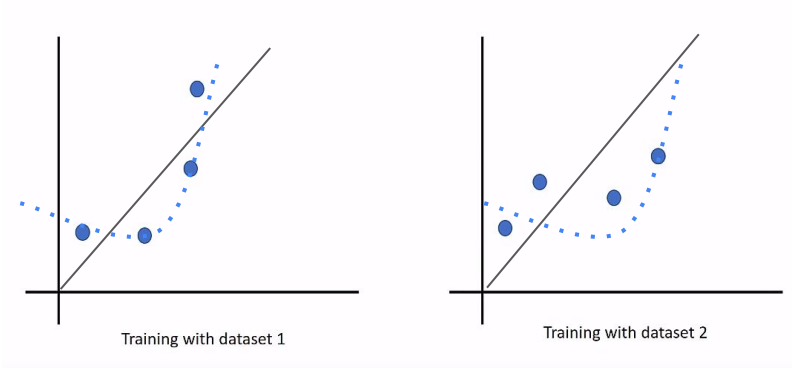
## Visualización (regresión)

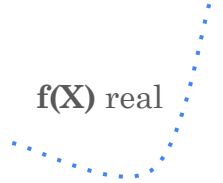
$$\text{Sesgo}\left[pred^{(i)}\right] = \text{E}\left[dif\left(pred^{(i)}, f(x^{(i)})\right)\right]$$

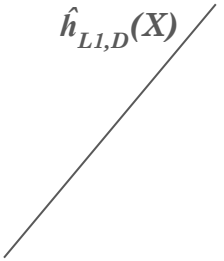
$$\text{Var}\left[pred^{(i)}\right] = \text{E}[dif\left(pred^{(i)}, \overline{pred}^{(i)}\right)^2]$$

**Algoritmo:**
Regresión Lineal

Sesgo: ??
Varianza: ??

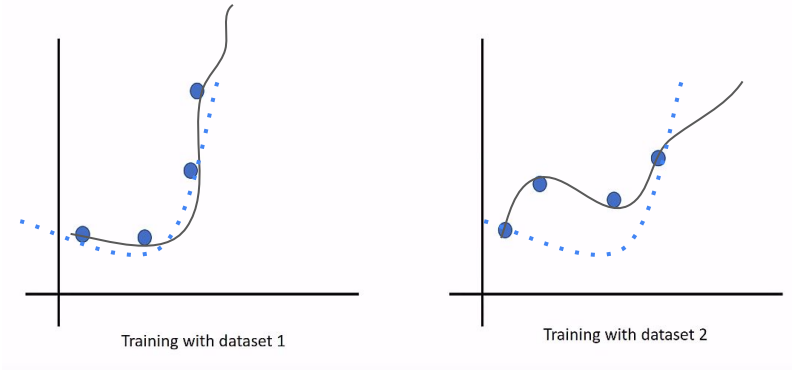


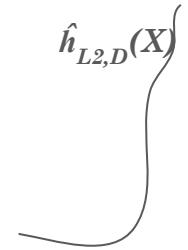




**Algoritmo:**
Regresión Polinómica

Sesgo: **??**
Varianza: ??



# Sesgo y Varianza

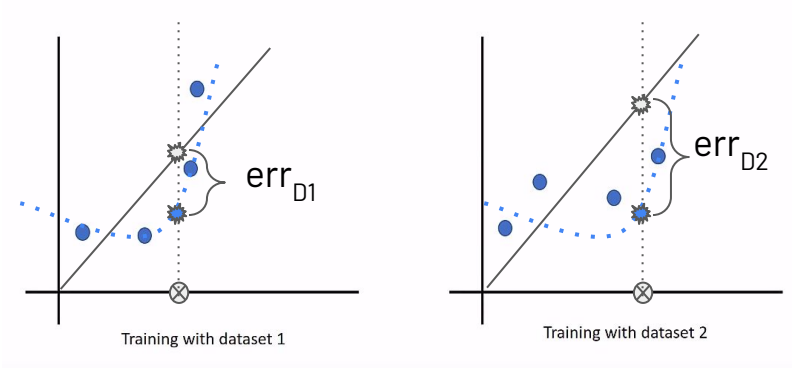
## Visualización (regresión)

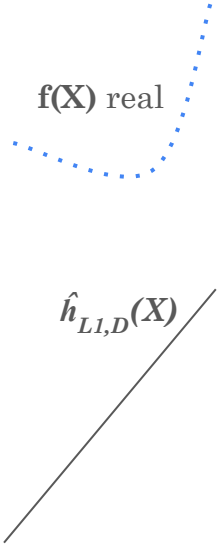
$$\text{Sesgo}\left[pred^{(i)}\right] = \text{E}\left[dif\left(pred^{(i)}, f(x^{(i)})\right)\right]$$

$$\text{Var}\left[pred^{(i)}\right] = \text{E}[dif\left(pred^{(i)}, \overline{pred}^{(i)}\right)^2]$$

**Algoritmo:**
Regresión Lineal

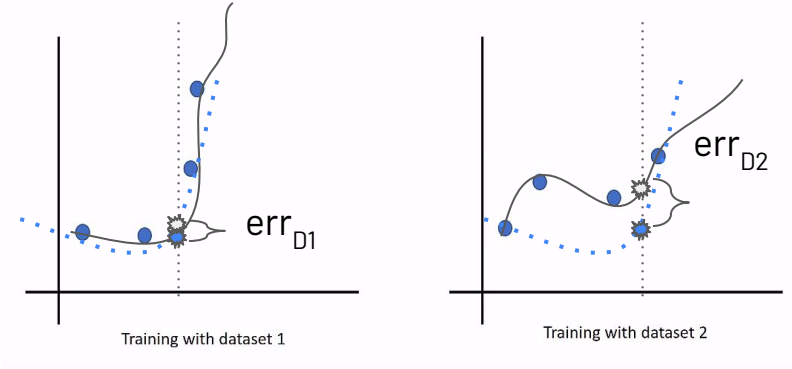
Sesgo: ??
Varianza: ??




**Algoritmo:**
Regresión Polinómica

Sesgo: ??
Varianza: ??



# Sesgo y Varianza

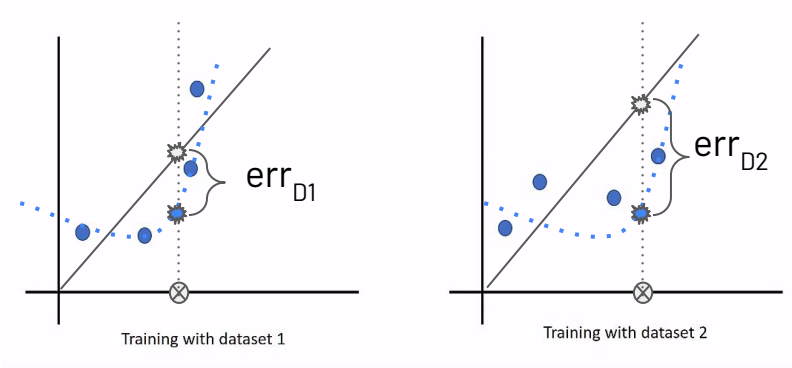
## Visualización (regresión)

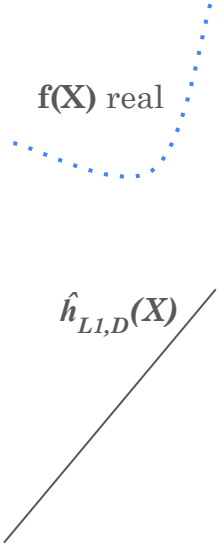
$$\text{Sesgo}\left[pred^{(i)}\right] = \text{E}\left[dif\left(pred^{(i)}, f(x^{(i)})\right)\right]$$

$$\text{Var}\left[pred^{(i)}\right] = \text{E}[dif\left(pred^{(i)}, \overline{pred}^{(i)}\right)^2]$$

**Algoritmo:**
Regresión Lineal

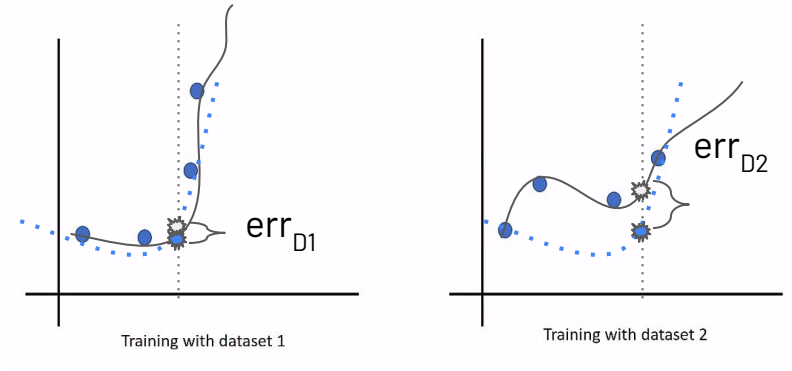
Sesgo: **Alto (Underfit)**
Varianza: ??

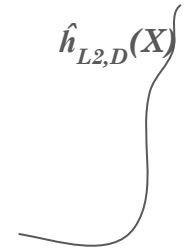



**Algoritmo:**
Regresión Polinómica

Sesgo: **Bajo**
Varianza: ??



# Sesgo y Varianza

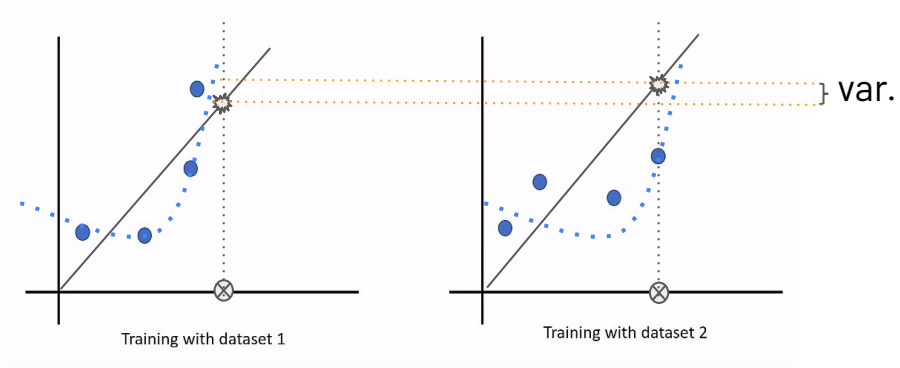
## Visualización (regresión)

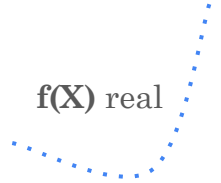
$$\text{Sesgo}\left[pred^{(i)}\right] = \text{E}\left[dif\left(pred^{(i)}, f(x^{(i)})\right)\right]$$

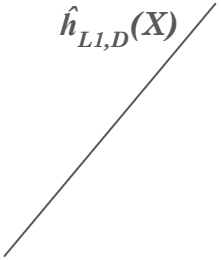
$$\text{Var}\left[pred^{(i)}\right] = \text{E}[dif\left(pred^{(i)}, \overline{pred}^{(i)}\right)^2]$$

**Algoritmo:**
Regresión Lineal

Sesgo: **Alto (Underfit)**
Varianza: ??

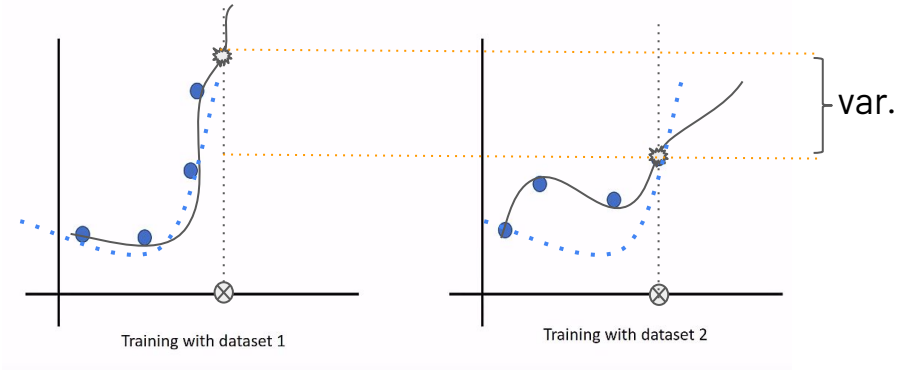


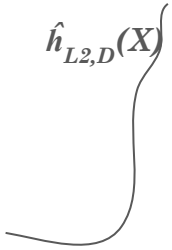




**Algoritmo:**
Regresión Polinómica

Sesgo: **Bajo**
Varianza: ??



# Sesgo y Varianza

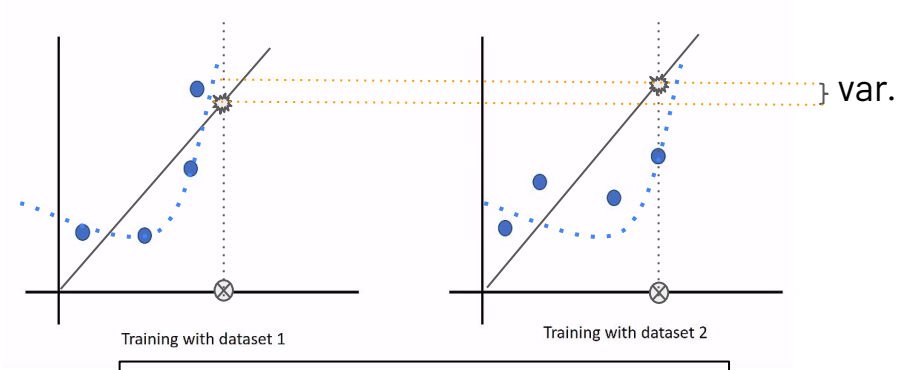
## Visualización (regresión)

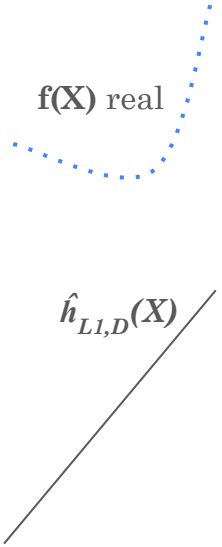
$$\text{Sesgo}\left[pred^{(i)}\right] = \text{E}\left[dif\left(pred^{(i)}, f(x^{(i)})\right)\right]$$

$$\text{Var}\left[pred^{(i)}\right] = \text{E}\left[dif\left(pred^{(i)}, \overline{pred}^{(i)}\right)^2\right]$$

**Algoritmo:**
Regresión Lineal

Sesgo: **Alto (Underfit)**
Varianza: **Baja**

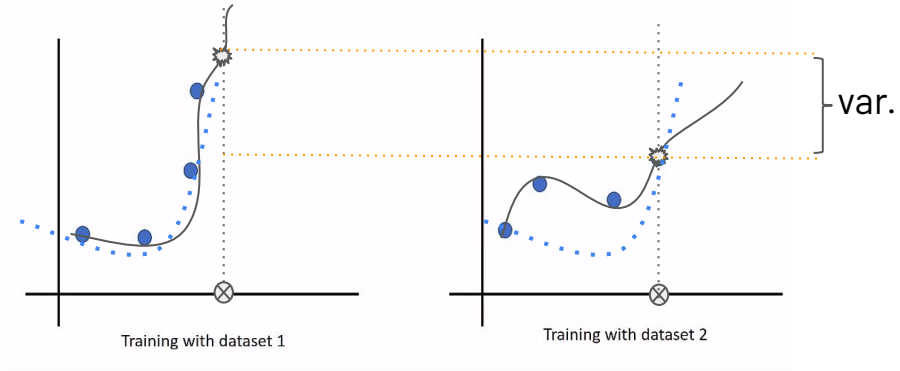


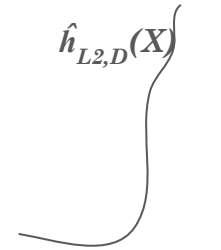




Notar que la varianza **no depende** de **f**.

**Algoritmo:**
Regresión Polinómica

Sesgo: **Bajo**
Varianza: **Alta (Overfit)**



# (fuera de programa)

## Tree Parzen Estimator (TPE)

# Búsqueda del mejor modelo

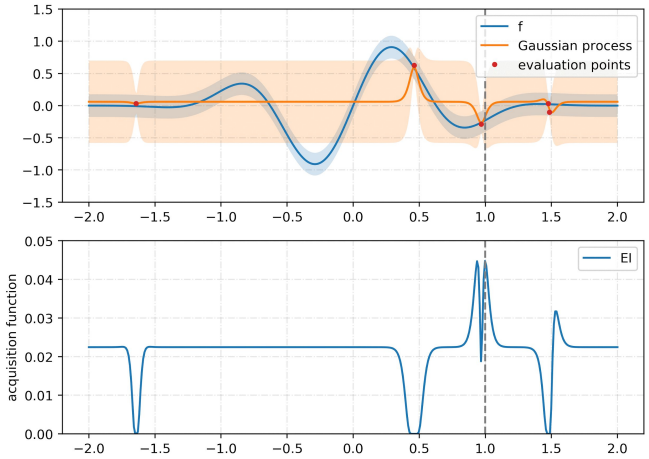
## Métodos de exploración de combinaciones: Optimización Bayesiana

Los métodos anteriores **no toman en cuenta** la información provista por **evaluaciones anteriores** para decidir cuál combinación probar a continuación.

Nos interesaría construir un modelo $P(s \mid h)$ en donde $\mathbf{s}$ es un puntaje de error (mientras más chico mejor), y $\mathbf{h}$ un valor para el hiperparámetro. Estimar $P(s \mid h)$, basado en observaciones anteriores, permite buscar en áreas prometedoras.

Métodos de esta pinta (aka **"Secuencial Model Based Optimization"**)

- Gaussian Process with Expected Improvement
- Random Forests (lo veremos más adelante)
- **Tree-structured Parzen Estimators (TPE)** (a continuación)




Fuente: https://techblog.criteo.com/hyper-parameter-optimization-algorithms-2fe447525903

[Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. (2011). "**Algorithms for hyper-parameter optimization**"]
[Bergstra, J., Yamins, D., & Cox, D. (2013, February). "**Making a science of model search: Hyperparameter optimization in hundreds of dimensions…**"]

# Búsqueda del mejor modelo

## Métodos de exploración de combinaciones: Tree Parzen Estimator (TPE)

Implementación: https://hyperopt.github.io/hyperopt/

Idea, modelar probabilidad de observar un hiperparámetro dado un score aproximando $P(h \mid s)$ y $P(s)$

Bayes

$$P(s \mid h) = \frac{P(h \mid s) \cdot P(s)}{P(h)}$$

En realidad, nos interesa modelar dos distribuciones:

$P(h \mid buen\_puntaje)$ (distr. de configuraciones exitosas)
Esta es la distribución de hiperparametros dado que el puntaje está por debajo de un cierto umbral.

$$P(h \mid s < s^{\star}_{\gamma})$$

γ = 0.2 devuelve s* tal que quedan 20% mejores combinaciones como "conf exitosas" : $P(s < s^{\star}_{\gamma}) = \gamma$

$P(h \mid mal\_puntaje)$ (distr. de configuraciones malas)
Esta es la distribución de hiperparametros dado que el puntaje está por encima del umbral.

$$P(h \mid s \geq s^{\star}_{\gamma})$$

Intuición: Un candidato "prometedor" se obtendrá a partir de muestrear $P(h \mid bueno)$. Entre todas las muestras, seleccionar la que maximice la "mejora esperada"

$$mejora\_esperada(h) \propto \frac{P(h \mid s < s^{\star}_{\gamma})}{P(h \mid s \geq s^{\star}_{\gamma})}$$

*Demo en paper [Begstra 2011]*



[Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. (2011). "**Algorithms for hyper-parameter optimization**"]

$$mejora\_esperada(h) \propto \frac{P(h \mid s < s^{\star}_{\gamma})}{P(h \mid s \geq s^{\star}_{\gamma})}$$

# Búsqueda del mejor modelo

## Métodos de exploración de combinaciones: Tree Parzen Estimator (TPE)

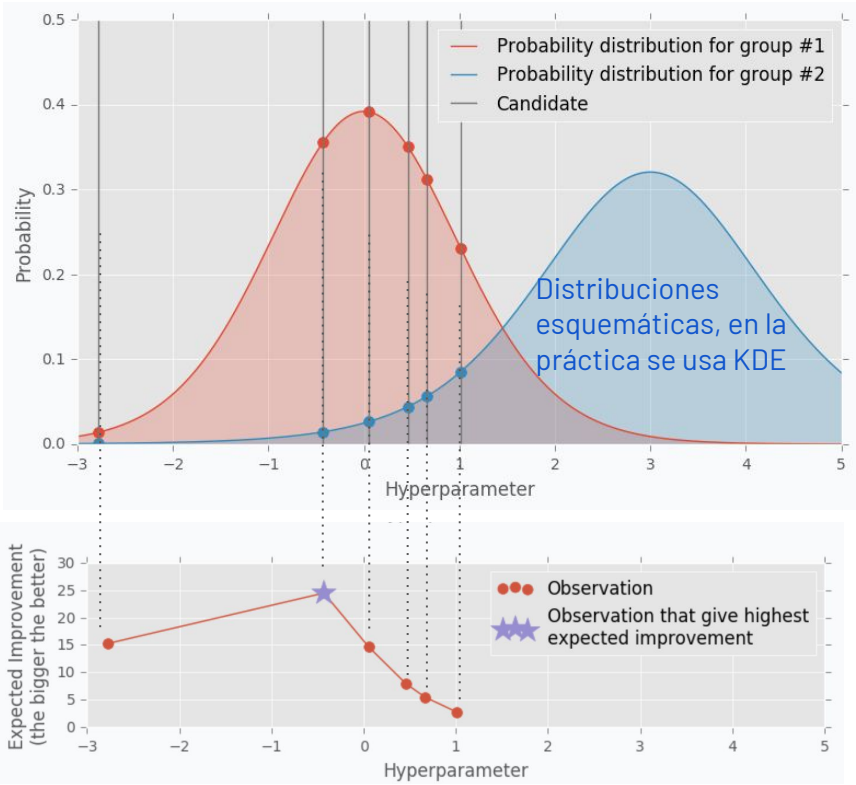
Idea, modelar probabilidad de observar un hiperparámetro dado un score aproximando $P(h \mid s)$ y $P(s)$

En realidad, nos interesa modelar dos distribuciones:

$P(h \mid buen\_puntaje)$ (distr. de configuraciones exitosas)
Esta es la distribución de hiperparametros dado que el puntaje está por debajo de un cierto umbral.

$P(h \mid mal\_puntaje)$ (distr. de configuraciones malas)
Esta es la distribución de hiperparametros dado que el puntaje está por encima del umbral.

Intuición: Un candidato "prometedor" se obtendrá a partir de muestrear $P(h \mid bueno)$. Entre todas las muestras, seleccionar la que maximice la "mejora esperada"






[Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. (2011). "**Algorithms for hyper-parameter optimization**"]

# Búsqueda del mejor modelo

## Métodos de exploración de combinaciones: Tree Parzen Estimator (TPE)

**Algorithm:** Algoritmo de Estimador Parzen Estructurado en Árboles (TPE)

1: Definir la función objetivo $f(h) = s$ que necesita ser minimizada.
2: Establecer el espacio inicial de búsqueda de hiperparámetros $H = \{(h_i; f(h_i))\}$.
3: **while** la condición de parada no se cumple **do**
4: Obtener el umbral $s^\star_\gamma$ (el percentil $\gamma$ de los puntajes)
5: Dividir las observaciones en dos conjuntos:

- $L$: Conjunto de observaciones con valores de pérdida bajos: $\{h \in H | f(h) < s^\star_\gamma\}$
- $G$: Conjunto de observaciones con valores de pérdida altos: $\{h \in H | f(h) \geq s^\star_\gamma\}$

6: Ajustar dos modelos:

- $l(h)$: Una PDF que modela $P(h \mid h \in L)$ utilizando KDE.
- $g(h)$: Una PDF que modela $P(h \mid h \in G)$ utilizando KDE.

7: Definir la función de adquisición a maximizar: $EI(h) = \frac{l(h)}{g(h)}$ (Mejora Esperada).
8: Muestrear de $l(h)$ un conjunto de configuraciones $H'$
9: Quedarse con la más prometedora $h^\star = \arg\max_{h \in H'} EI(h)$.
10: Evaluar la función objetivo con la nueva configuración: $s' = f(h')$.
11: Actualizar las observaciones con el nuevo par $H \bigoplus (h'; s')$.
12: **end while**
13: Devolver la mejor configuración de hiperparámetros encontrada.



[Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. (2011). "**Algorithms for hyper-parameter optimization**"]