

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación



IIC2613 - Inteligencia Artificial

Representación de datos y generalización

Hans Löbel


Dpto. Ingeniería de Transporte y Logística
Dpto. Ciencia de la Computación

Algoritmos y modelos de ML trabajan sobre datos **multidimensionales -> vectores**

- Cada dato es caracterizado por una serie de **características = features = mediciones = atributos = variables**.
- La cantidad de **features** define la **dimensionalidad** del dato o vector.
- El espacio donde viven los datos se conoce como **espacio de características** (*feature space*).

California Housing Prices Data (5 new features!)

Median house prices for California districts derived from the 1990 census.



[Data Card](#) [Code \(25\)](#) [Discussion \(5\)](#) [Suggestions \(0\)](#)

About Dataset

Context

This is the dataset is a modified version of the California Housing Data used in the paper [Pace, R. Kelley, and Ronald Barry. "Sparse spatial autoregressions." Statistics & Probability Letters 33.3 \(1997\): 291-297.](#) . It serves as an excellent introduction to implementing machine learning algorithms because it requires rudimentary data cleaning, has an easily understandable list of variables and sits at an optimal size between being too toyish and too cumbersome.

The data contains information from the 1990 California census. So although it may not help you with predicting current housing prices like the Zillow Zestimate dataset, it does provide an accessible introductory dataset for teaching people about the basics of machine learning.

Modifications with respect to the original data

This dataset includes 5 extra features defined by me: "Distance to coast", "Distance to Los Angeles", "Distance to San Diego", "Distance to San Jose", and "Distance to San Francisco". These extra features try to account for the distance to the nearest coast and the distance to the centre of the largest cities in California.

The distances were calculated using the Haversine formula with the Longitude and Latitude:

Usability

10.00

License

Data files © Original Authors

Expected update frequency

Never

Tags

Earth and Nature

Social Science

United States

Regression

Cities and Urban Areas

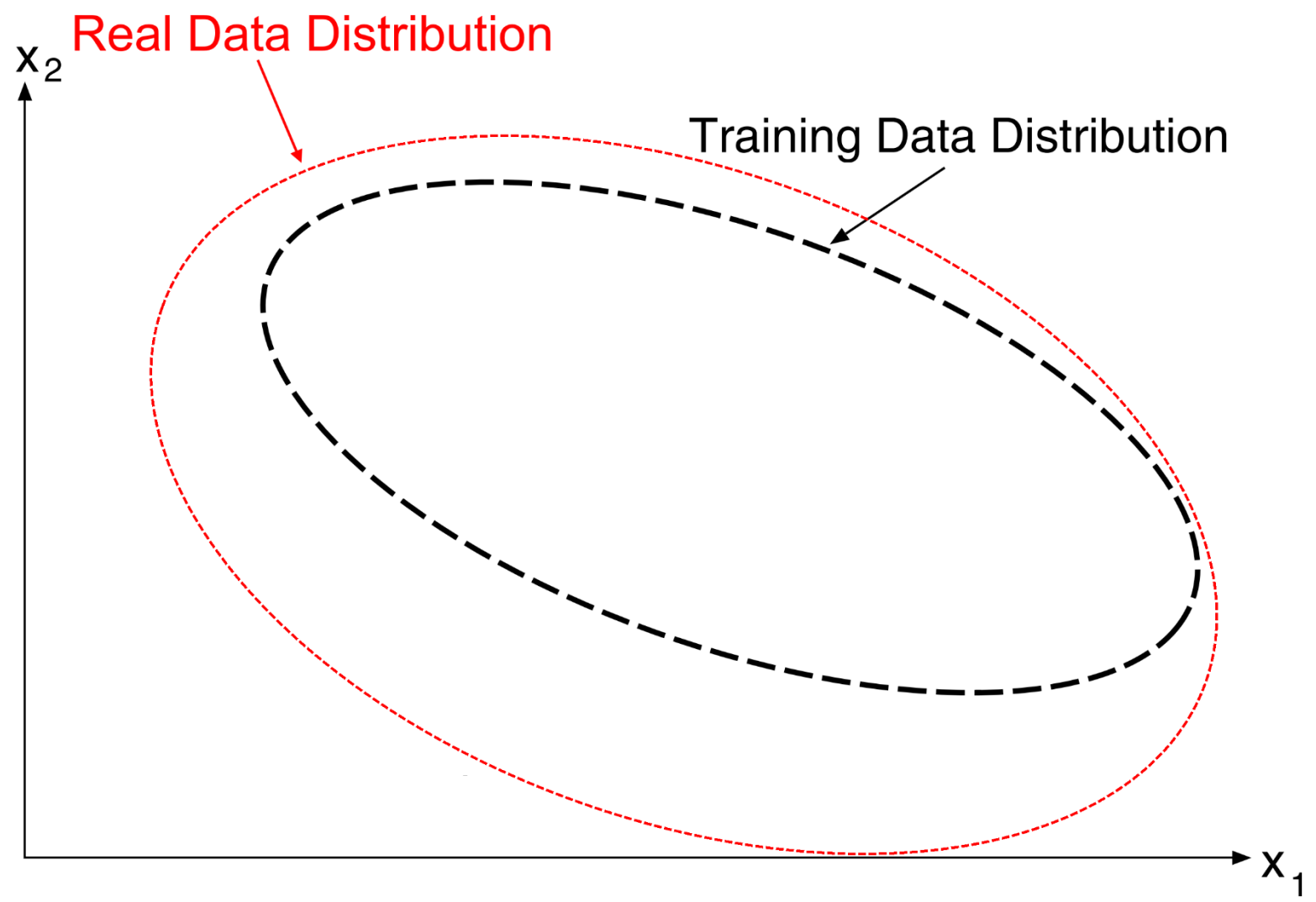
Housing

Urban Planning

Para **entrenar** = ajustar = calibrar un modelo,
se utiliza el **conjunto de entrenamiento**

Identificador	Vector de características			Etiqueta
Ciudad	Población	Superficie (km²)	Densidad poblacional por km²	Ingreso anual promedio per cápita en dólares
Metropolis	3.500.000	600	5.833	\$45.000
Rivertown	1.200.000	150	8.000	\$35.000
Coastline	2.800.000	300	9.333	\$40.000
Highland	900.000	350	2.571	\$30.000
Greenfield	600.000	400	1.500	\$28.000
Sunnyside	750.000	500	1.500	\$32.000
Frostville	300.000	200	1.500	\$27.000
Eastbank	1.100.000	250	4.400	\$36.000
Westwood	2.300.000	450	5.111	\$42.000
Clearwater	500.000	180	2.777	\$29.000

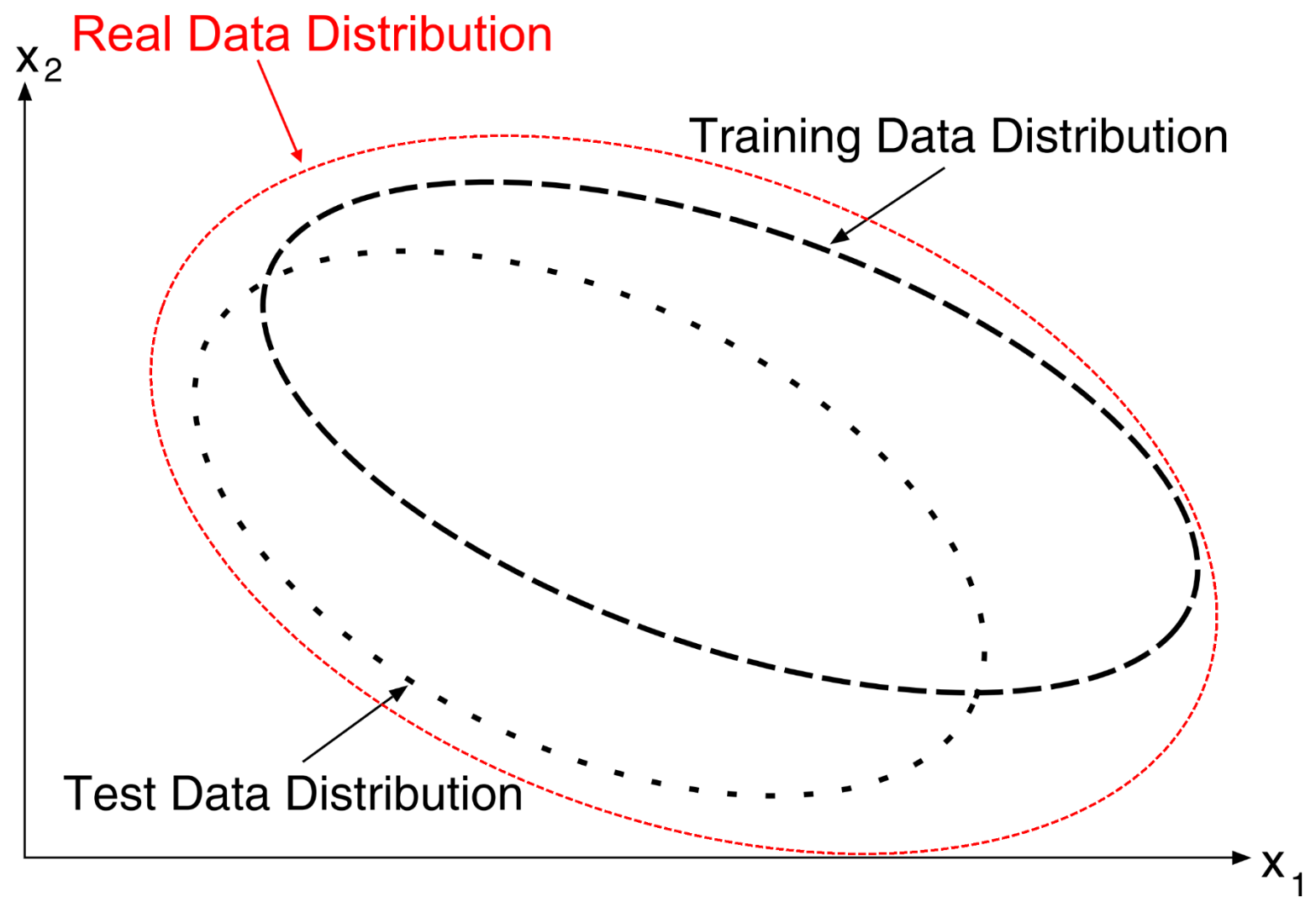
Cada dato (fila) del set de entrenamiento, puede considerarse como un **vector** en el espacio de características.

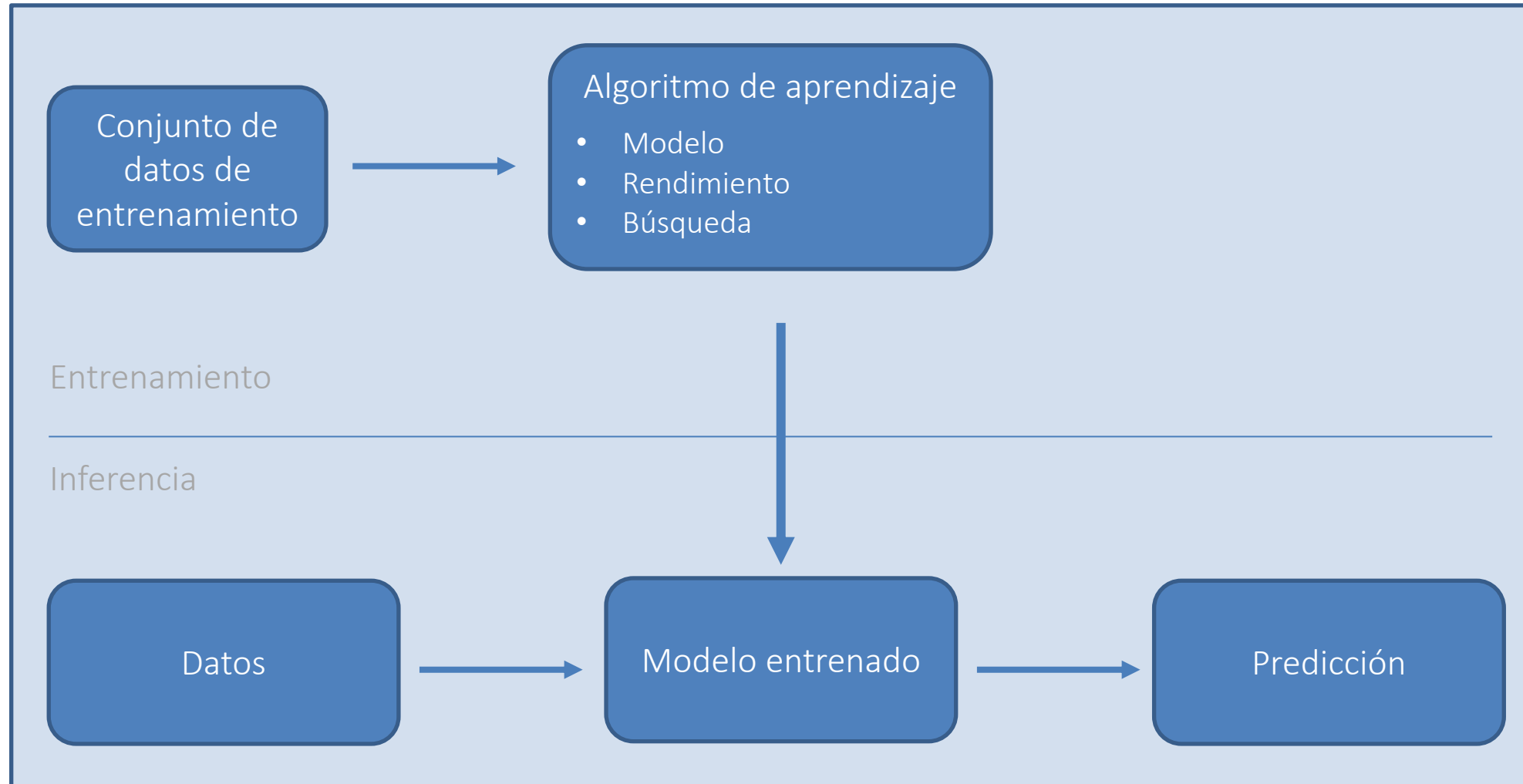


¿Cómo evaluamos la capacidad de **generalización**
(aprendizaje inductivo) del modelo?

	Ciudad	Población	Superficie (km²)	Densidad poblacional por km²	Ingreso anual promedio per cápita en dólares
Entrenamiento	Metropolis	3.500.000	600	5.833	\$45.000
	Rivertown	1.200.000	150	8.000	\$35.000
	Coastline	2.800.000	300	9.333	\$40.000
	Highland	900.000	350	2.571	\$30.000
	Greenfield	600.000	400	1.500	\$28.000
	Sunnyside	750.000	500	1.500	\$32.000
	Frostville	300.000	200	1.500	\$27.000
	Eastbank	1.100.000	250	4.400	\$36.000
	Westwood	2.300.000	450	5.111	\$42.000
	Clearwater	500.000	180	2.777	\$29.000
Test	Ironforge	800.000	190	4.211	?
	Lakeview	950.000	220	4.318	
	Cedarwood	410.000	300	1.367	
	Newhaven	1.500.000	280	5.357	
	Pinecrest	670.000	330	2.030	
	Stonebridge	390.000	210	1.857	

Etiquetas del set de test deben ser completamente desconocidas al momento de entrenar





(Interludio práctico) En este curso usaremos scikit-learn

- scikit-learn es el módulo para ML más conocido y utilizado en Python.
- Su principal atractivo es una interfaz limpia, uniforme y simple, que facilita la exploración y permite la integración con otros paquetes, como Pandas.
- Posee además de una completa documentación en línea (<https://scikit-learn.org/>).



Datos son representados por una *matriz de features* y un *vector objetivo/de etiquetas*.

- Las características de los ejemplos se almacenan en una matriz de *features* (**X**), de tamaño $[n_samples, n_features]$ (esta matriz puede ser un **DataFrame** de Pandas, por ejemplo).
- El vector objetivo (**y**) contiene el valor a predecir para cada ejemplo y tiene tamaño $[n_samples, 1]$ (este vector puede ser una **Series** de Pandas).
- Y eso es todo...

Feature Matrix (X)

n_features →

n_samples

[illegible]

Target Vector (y)

t—n_samples

[illegible]

Interfaz para usar modelos/algoritmos

En general, un caso de uso típico en scikit-learn es como el siguiente:

1. Elegir el modelo adecuado, importando la clase correspondiente desde *sklearn*.
2. Obtener o generar matriz *X* y vector *y*.
3. Entrenar el modelo llamando al *fit(X, y)*.
4. Aplicar el modelo al set de test, usando el método *predict()*.
5. Evaluar el rendimiento con algunas de las métricas disponibles en *sklearn.metrics*.

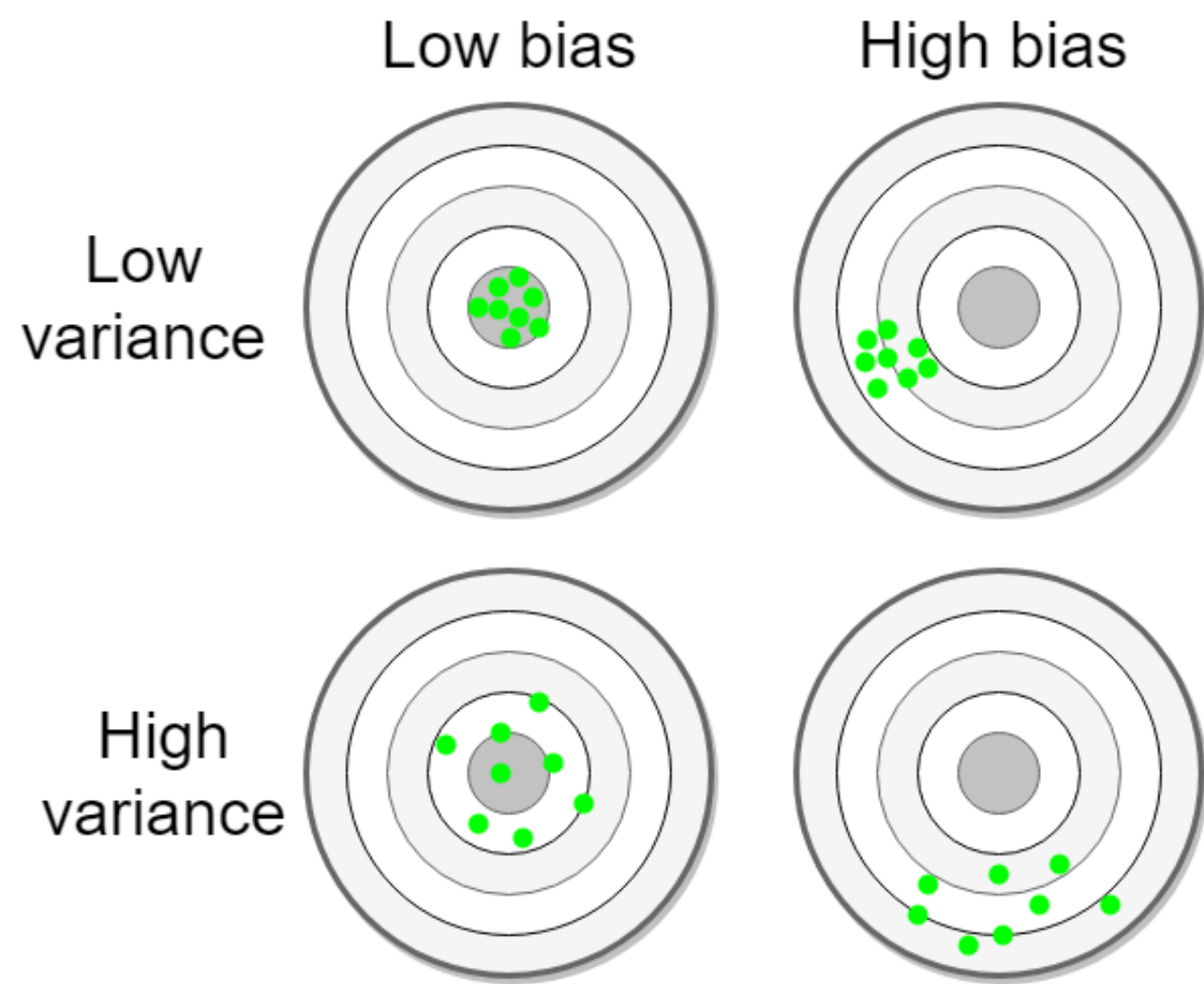
Interfaz para usar modelos/algoritmos

- La interfaz de scikit-learn se basa en los siguientes conceptos principales:
 - Consistente: todos los modelos comparten una interfaz con unas pocas funciones.
 - Sucinta: solo usa clases propias para los algoritmos. Para todo el resto utiliza formatos estándares (datos en DataFrame, por ejemplo).
 - Útil: los parámetros por defecto son útiles para estimar adecuadamente los modelos.
- En resumen, requiere muy poco esfuerzo utilizarla y obtener resultados rápidamente.



A pesar de ser clave, **los datos** no lo son todo

- En general, los algoritmos de aprendizaje viven y mueren por el set de entrenamiento.
- Lamentablemente, tener un buen set de entrenamiento, **no asegura siempre tener buena generalización**.
- Poder de representación o complejidad del modelo predictivo pasa a ser también un tema central.
- El porqué de esto está dado por un problema llamado **bias-variance tradeoff**



Bias-variance tradeoff se da de **forma natural** en ML

$$y = f(x) + \varepsilon, \varepsilon \sim N(0, \sigma^2)$$

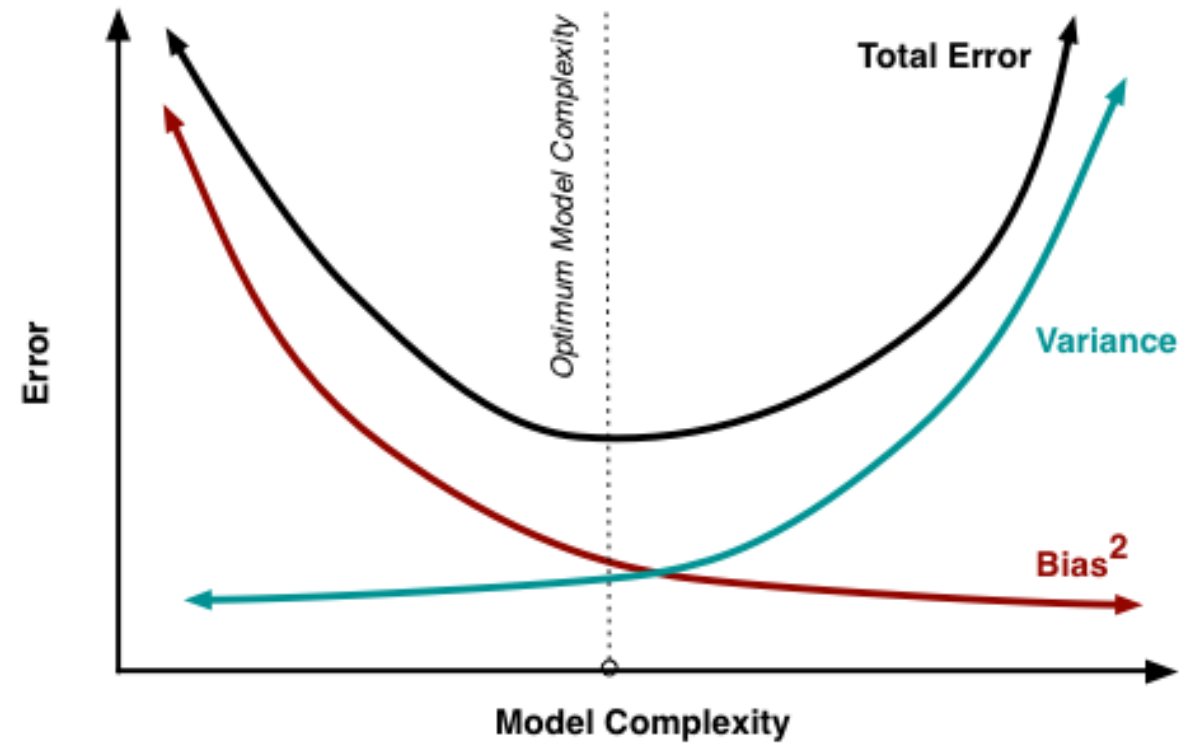
$$f(x) \approx \hat{f}(x)$$

$$Err(x) = E \left[\left(y - \hat{f}(x) \right)^2 \right]$$

$$Err(x) = \left(f(x) - E[\hat{f}(x)] \right)^2 + E \left[\left(\hat{f}(x) - E[\hat{f}(x)] \right)^2 \right] + \sigma^2$$

$$Err(x) = Bias^2 + Varianza + Error\ irreducible$$

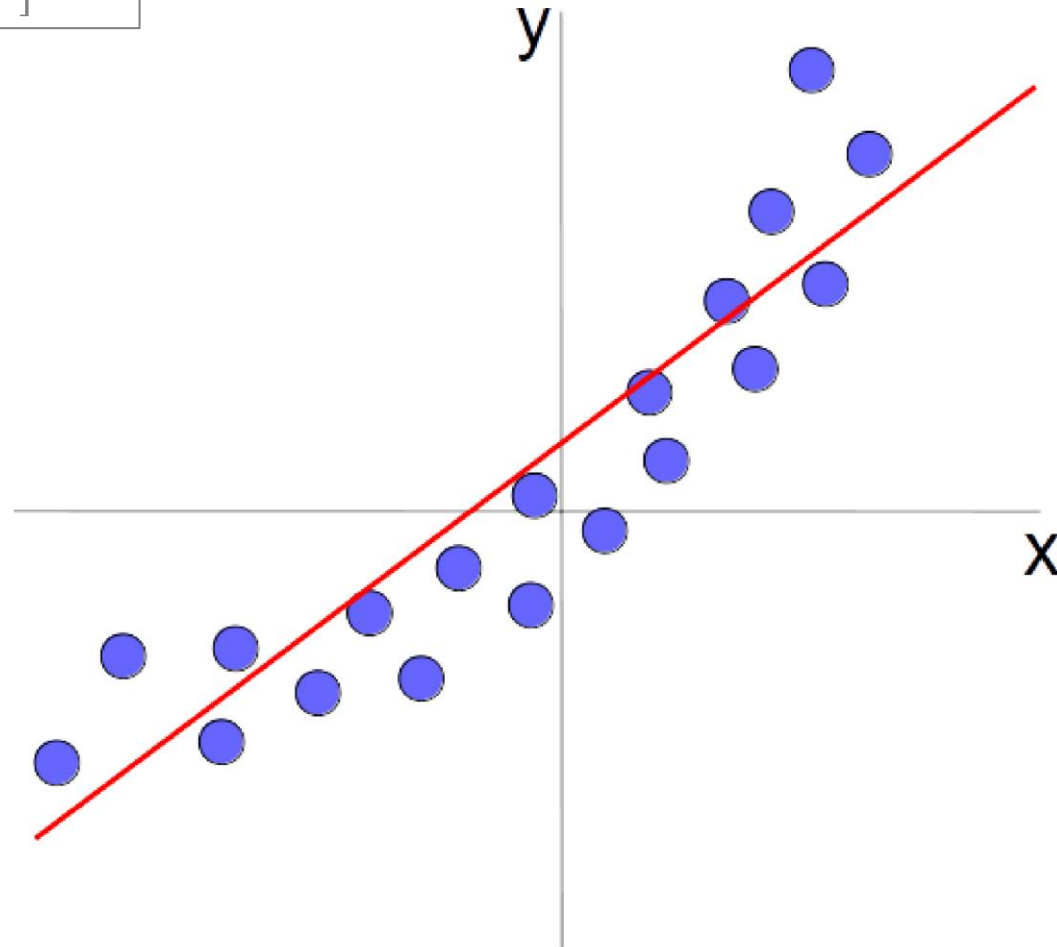
Complejidad del modelo es lo que permite capturar el *bias-variance tradeoff*



$$Err(x) = Bias^2 + Varianza + Error\ irreducible$$

$$Err(x) = E \left[\left(Y - \hat{f}(x) \right)^2 \right]$$

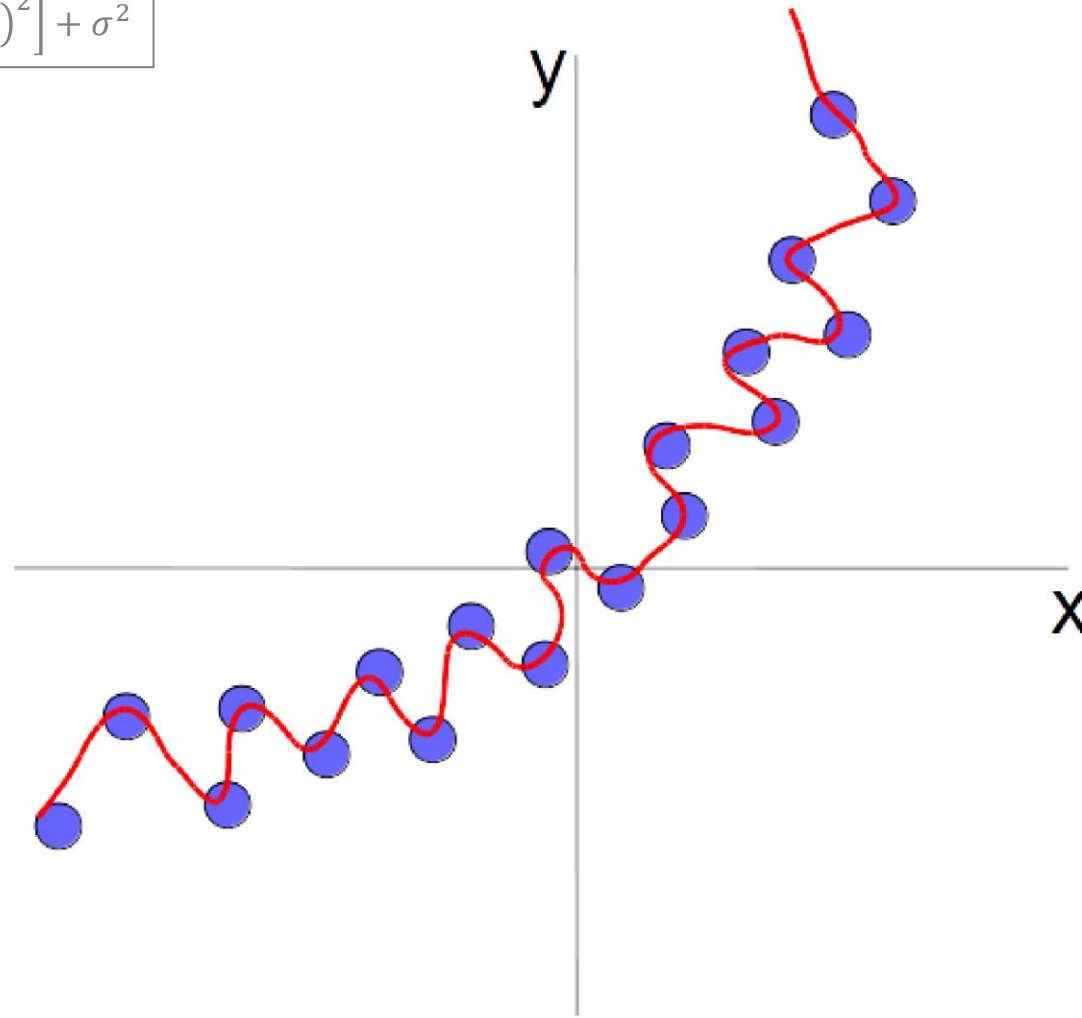
$$Err(x) = \left(E[\hat{f}(x)] - f(x) \right)^2 + E \left[\left(\hat{f}(x) - E[\hat{f}(x)] \right)^2 \right] + \sigma^2$$



Subajuste: modelo es demasiado simple para capturar el comportamiento de los datos (*underfitting, alto sesgo*)

$$Err(x) = E \left[\left(Y - \hat{f}(x) \right)^2 \right]$$

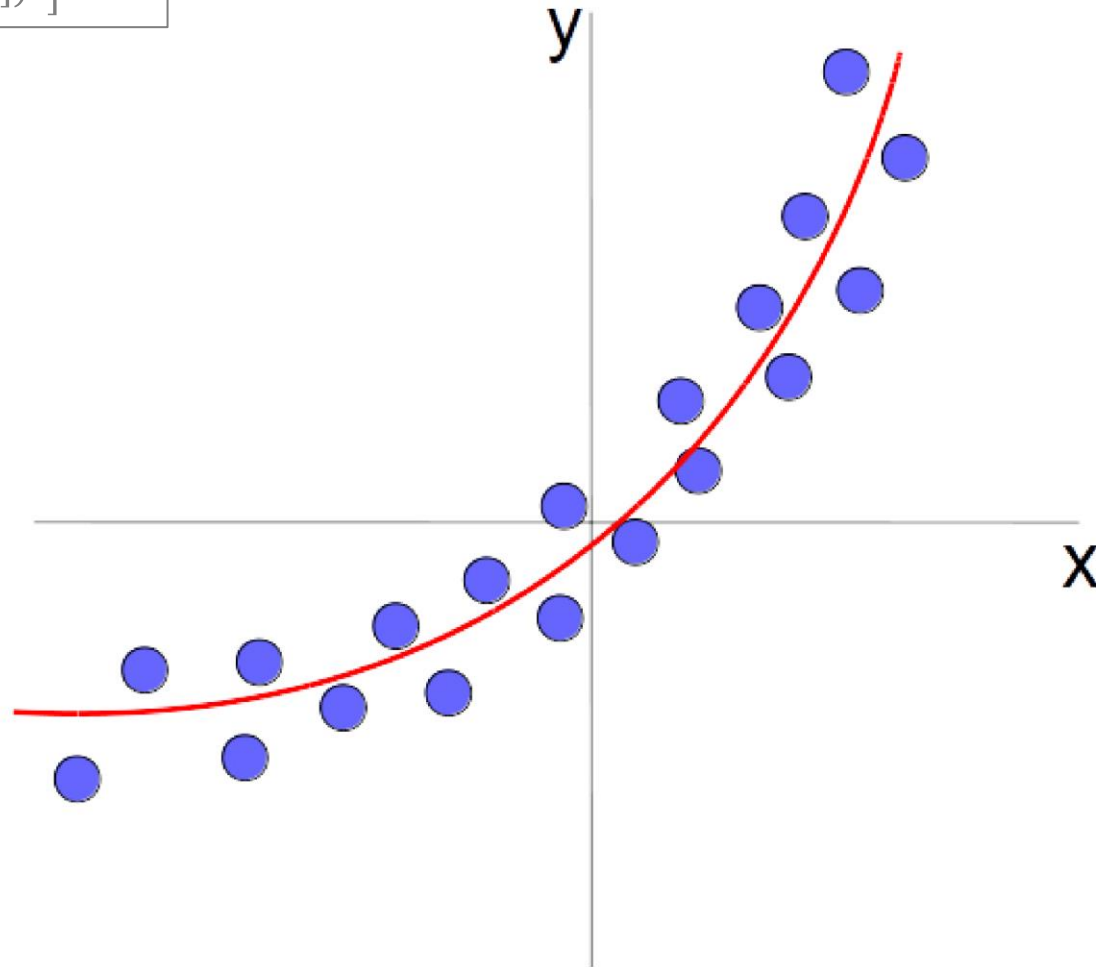
$$Err(x) = \left(E[\hat{f}(x)] - f(x) \right)^2 + E \left[\left(\hat{f}(x) - E[\hat{f}(x)] \right)^2 \right] + \sigma^2$$



Sobreajuste: modelo es muy complejo, y captura hasta el ruido presente en los ejemplos (*overfitting, alta varianza*)

$$Err(x) = E \left[\left(Y - \hat{f}(x) \right)^2 \right]$$

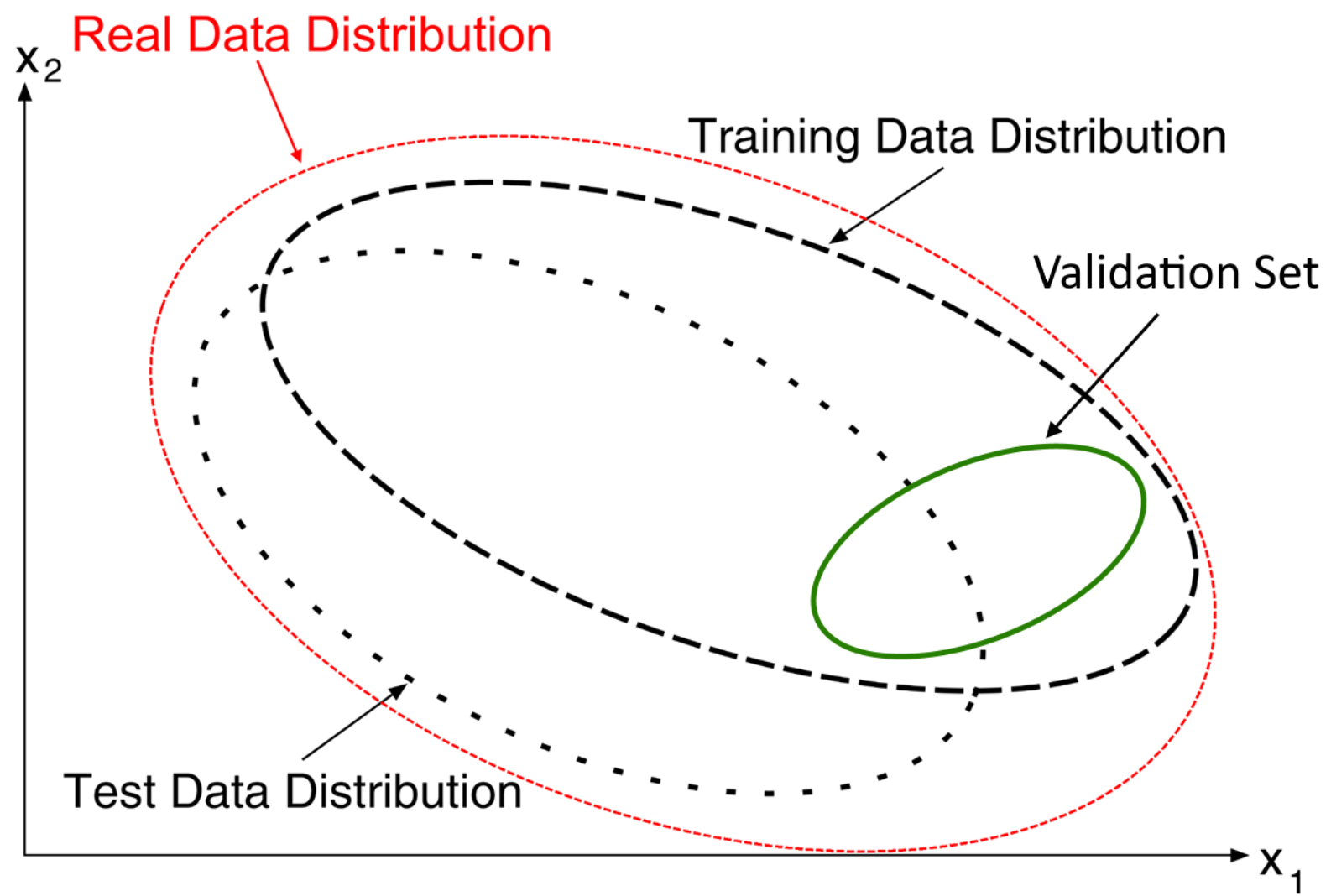
$$Err(x) = \left(E[\hat{f}(x)] - f(x) \right)^2 + E \left[\left(\hat{f}(x) - E[\hat{f}(x)] \right)^2 \right] + \sigma^2$$



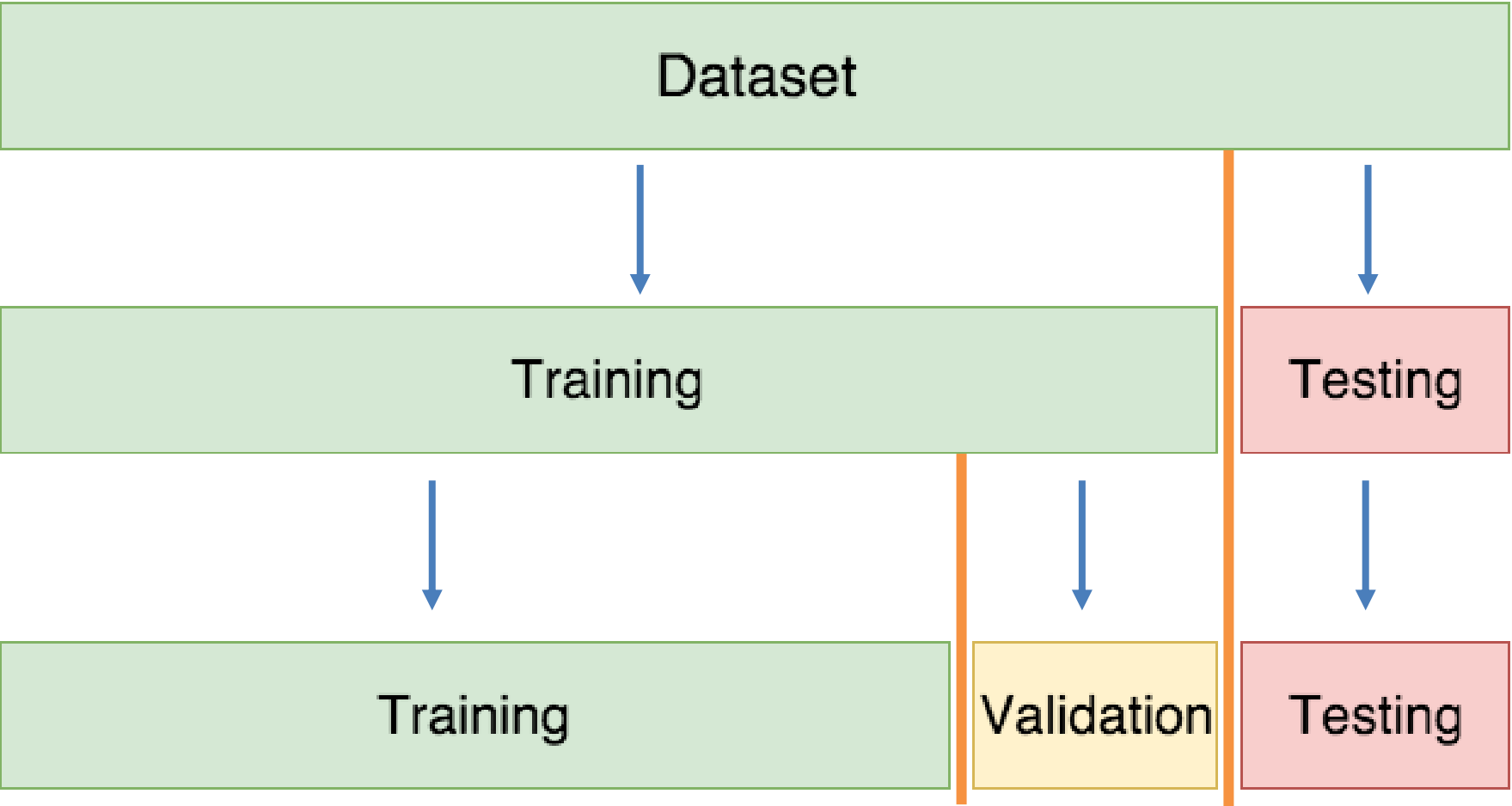
Modelo tiene la complejidad necesaria para capturar los patrones relevantes, **controlando sesgo y varianza**.

¿Cómo podemos enfrentar esto?

- Un mecanismo típico es utilizar un conjunto de validación para evaluar el rendimiento.
- El conjunto de validación es una pequeña parte del conjunto de entrenamiento, que no se usa para entrenar inicialmente.

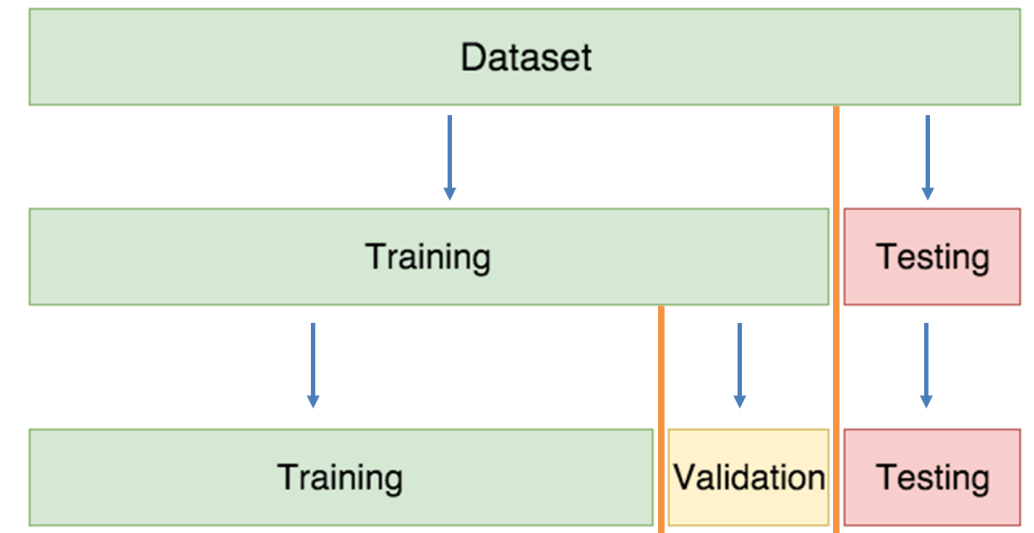


Otra forma de verlo, con conjuntos disjuntos

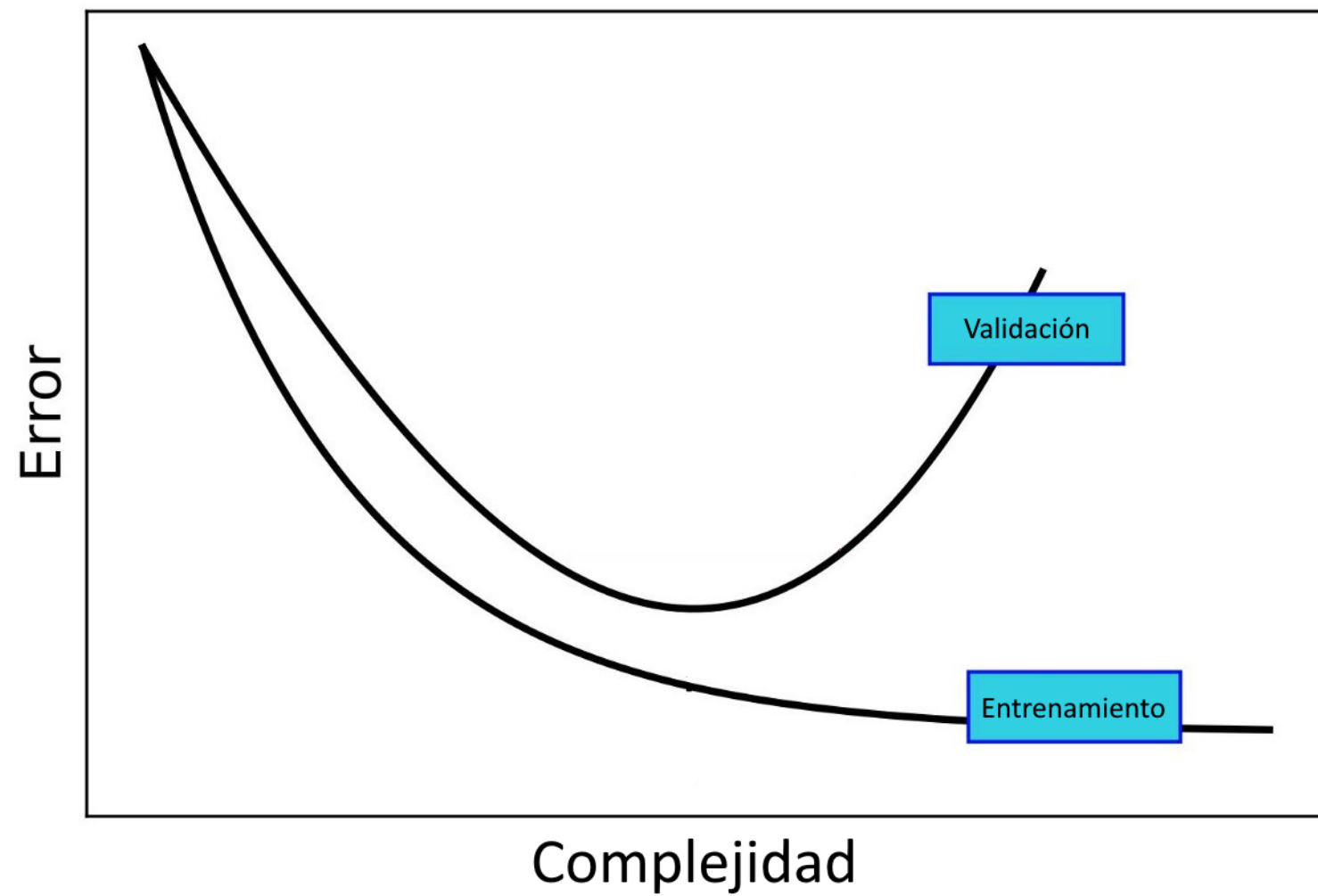


¿Cómo podemos enfrentar esto?

- Un mecanismo típico es utilizar un set de validación para evaluar el rendimiento.
- El set de validación es una pequeña parte del set de entrenamiento, que no se usa para entrenar inicialmente.
- Se entrenan distintos modelos en el nuevo conjunto de entrenamiento y se evalúan en el de validación.
- El modelo con mejor rendimiento en validación es el elegido.

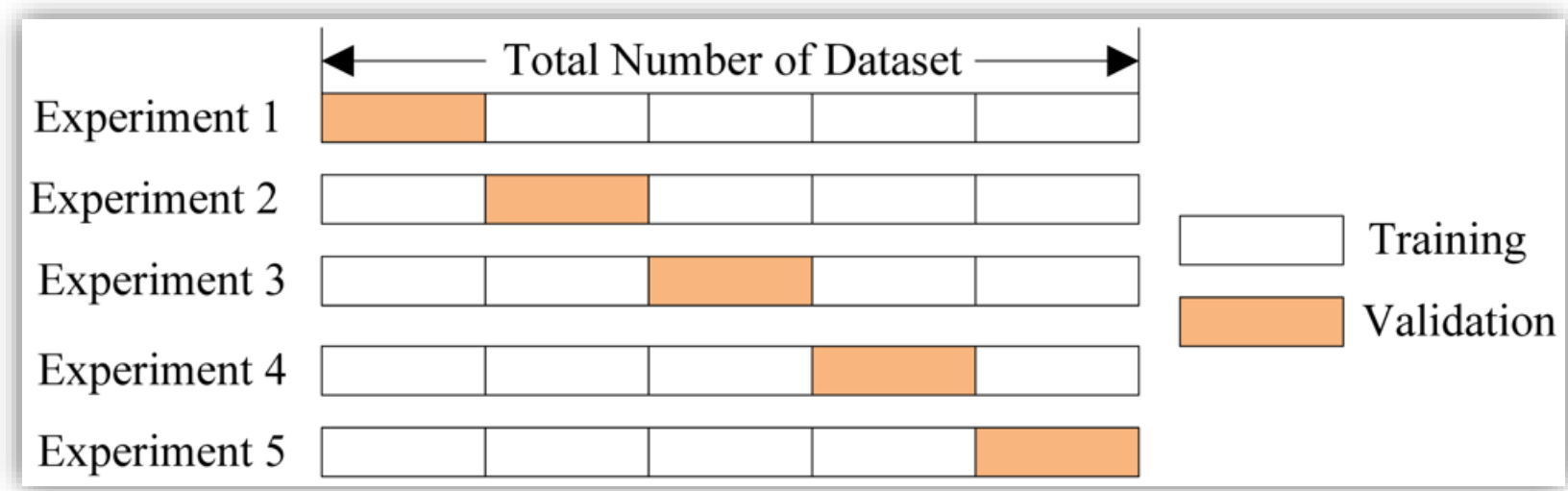


En general, error en validación baja y luego sube



Una mejor opción, pero no siempre factible,
es utilizar validación cruzada

- Utilizar múltiples (y distintos) conjuntos de entrenamiento y validación, en base al conjunto original de entrenamiento.
- Permite caracterizar mejor el rendimiento real, pero puede ser altamente costosa en tiempo.



Cuáles son los conceptos centrales de estas primeras clases

- Programación directa vs aprendizaje para resolver problemas.
- Diferenciación entre visión de IA clásica y ML.
- Tipos de aprendizaje.
- Esquema general de un sistema de ML.
- Conjuntos de datos involucrados: entrenamiento, test, validación.
- Bias-Variance Tradeoff: overfitting y underfitting

Lecturas opcionales de profundización

- Dos libros muy útiles para el curso son *Python Data Science Handbook* y *The Hundred-Page Machine Learning Book*.
- “Funes el Memorioso”, de Jorge Luis Borges.

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación



IIC2613 - Inteligencia Artificial

Representación de datos y generalización

Hans Löbel

Dpto. Ingeniería de Transporte y Logística
Dpto. Ciencia de la Computación