UNIVERSIDAD
COMPLUTENSE
MADRID

Desarrollo de Aplicaciones y Sistemas Inteligentes (DASI)

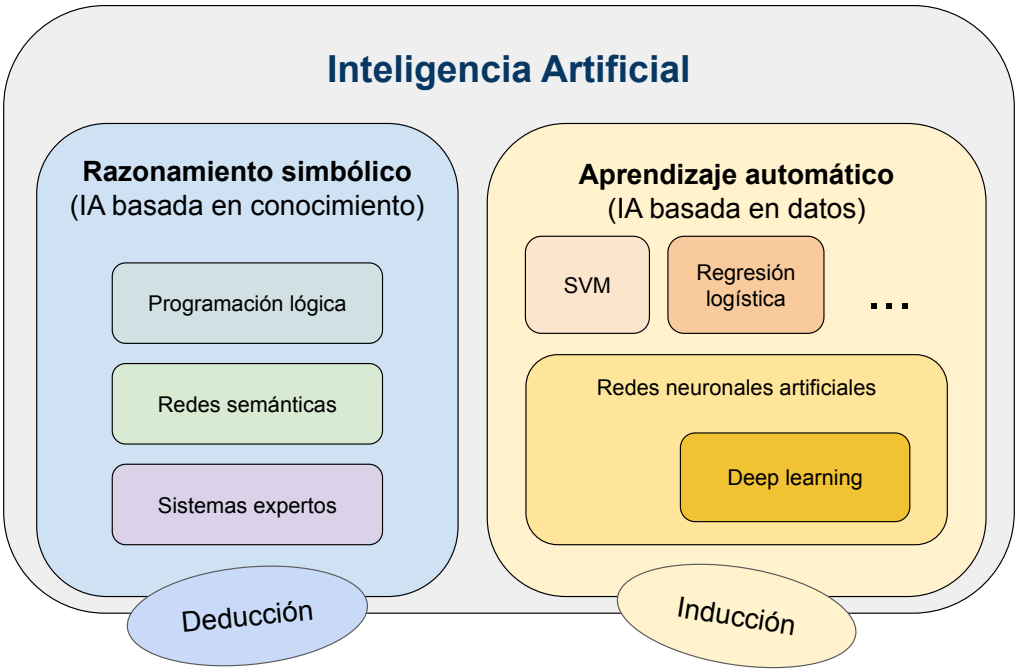
Introducción al Aprendizaje Automático

Juan Pavón Mestras
Dep. Ingeniería del Software e Inteligencia Artificial UCM

Tipos de sistemas inteligentes

- **Sistemas basados en conocimiento (IA simbólica)**
  - La inteligencia se puede reducir a un mero problema de manipulación de símbolos, cuyas relaciones se pueden organizar en estructuras como listas, jerarquías o redes
  - El conocimiento se representa de manera explícita y formal (mediante símbolos) para que pueda ser procesada por un algoritmo de razonamiento e inferir soluciones a problemas o preguntas del dominio
  - Ejemplos: Sistemas expertos, algoritmos de búsqueda, inferencia, ontologías, planes
- **Sistemas dirigidos por datos** (***Machine learning***)
  - No hay una representación explícita del conocimiento
  - El sistema aprende a partir de ejemplos o de la experiencia en el uso del sistema
  - Los datos observados proporcionan información incompleta sobre fenómenos y los algoritmos de aprendizaje intentan generalizar la información para hacer predicciones sobre fenómenos
  - Ejemplos: Redes neuronales artificiales, deep learning, SVM, aprendizaje bayesiano, etc.

Métodos de IA



Deducción vs. inducción

- Razonamiento **deductivo**
  - Infiere hechos/verdades particulares a partir de una verdad general, basado en un sistema de lógica
  - El razonamiento será válido o inválido

    Todos los humanos son mortales
    Sócrates es humano
    ⇒ Sócrates es mortal

- Razonamiento **inductivo**
  - Las premisas sirven de evidencia (fundamento) para concluir algo
  - Se infieren verdades generales a partir de verdades particulares
  - El razonamiento no es válido o inválido, solo probable (mejor o peor)

    Sócrates es humano y mortal
    Platón es humano y mortal
    Aristóteles es humano y mortal
    *⇒ **Probablemente**, todos los humanos son mortales*

Tipos de problemas que se resuelven con ML

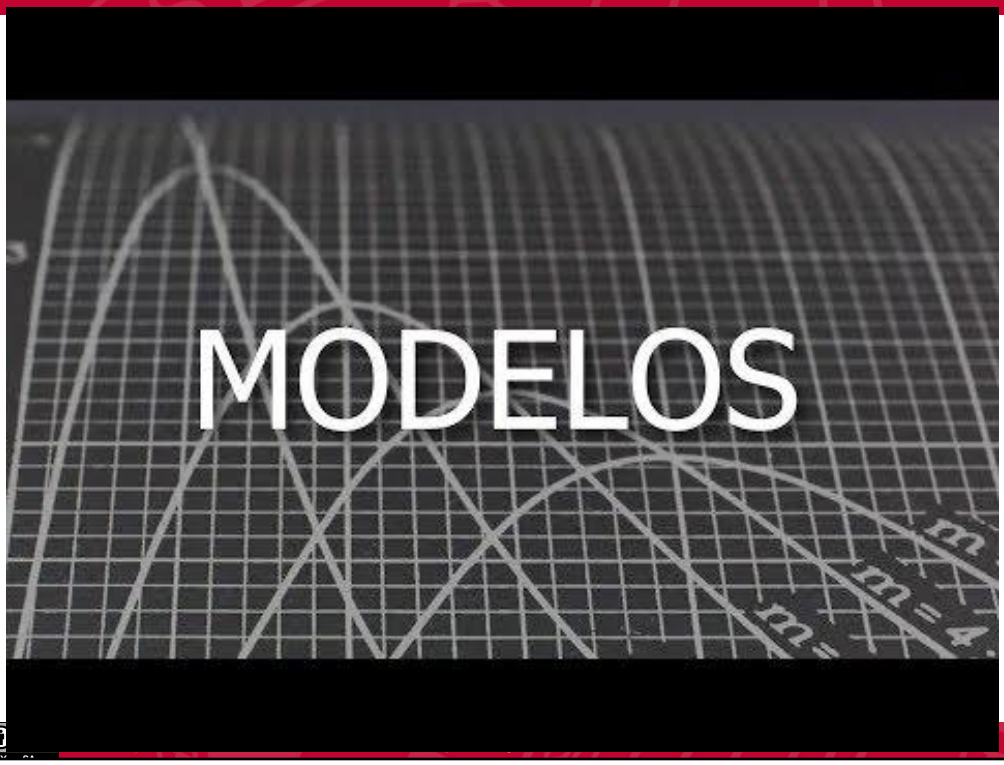
- **Clasificación**: indicar a qué clase predefinida pertenece un objeto
- **Agrupamiento (*clustering*)**: dividir un conjunto de objetos heterogéneos en grupos homogéneos
- **Diagnóstico**: inferir problemas en el funcionamiento y sugerir soluciones
- **Selección**: recomendar la mejor opción de entre varias alternativas
- **Predicción**: predecir el comportamiento futuro en base al pasado
- **Optimización**: mejorar una solución hasta obtener la óptima
- **Control**: regular el comportamiento de un objeto en tiempo real de acuerdo a una serie de requisitos

Aprendizaje automático

- La principal tarea de un sistema de aprendizaje automático es
  - Construir **modelos**
  - Que **aprendan** de **datos** históricos
  - Para poder hacer **predicciones** sobre nuevos datos de entrada
- Para ello es necesario definir una **función de evaluación:**
  Función de coste (*cost function*) o de **pérdida** (*loss function*)
  - Mide lo buenos que son los modelos aprendidos

  ⇒ El aprendizaje automático es un problema de optimización con el propósito de aprender de la forma más eficiente

Construir MODELOS - https://youtu.be/Sb8XVheowVQ



Tipos de aprendizaje máquina

- Aprendizaje **supervisado**
  - Se le presenta al sistema un conjunto de entrenamiento (pares de problemas *entrada-solución*) ⇒ **Datos etiquetados**
    - O se corrigen las soluciones que genera a partir de las entradas
  - El sistema almacena y generaliza estas soluciones
  - Requiere preparar (*etiquetar*) las entradas (y soluciones) apropiadas
  - Suele tener problemas de adaptación en entornos cambiantes
    ⇒ Necesita nuevo entrenamiento
- Aprendizaje **no supervisado**
  - El sistema no recibe información (*feedback*) acerca de la salida esperada
    ⇒ **Datos no etiquetados**
    - Aprende de su propia observación
    - A partir de las propiedades de las entradas realiza una agrupación o caracterización (clasificación, clustering) de las entradas según la similitud de sus propiedades
  - Le permite adaptarse mejor a entornos cambiantes
  - Pero las soluciones aprendidas pueden no ser las deseadas
    - Depende de la experiencia con la que aprendió

Más tipos de aprendizaje máquina

- Aprendizaje **semi-supervisado**
  - Una combinación de los dos anteriores
  - Utiliza ejemplos clasificados (etiquetados) y no clasificados (sin etiquetar)
  - Útil cuando algunos ejemplos no están clasificados o lo están erróneamente
- Aprendizaje **por refuerzo**
  - El sistema se adapta a condiciones dinámicas del entorno persiguiendo un objetivo
  - El agente aprende **interaccionando con el entorno** y evaluando las consecuencias de sus acciones (recompensas o castigos)
  - Proceso de prueba y error, reforzando aquellas acciones que reciben una respuesta positiva en el mundo

Tipos de aprendizaje automático

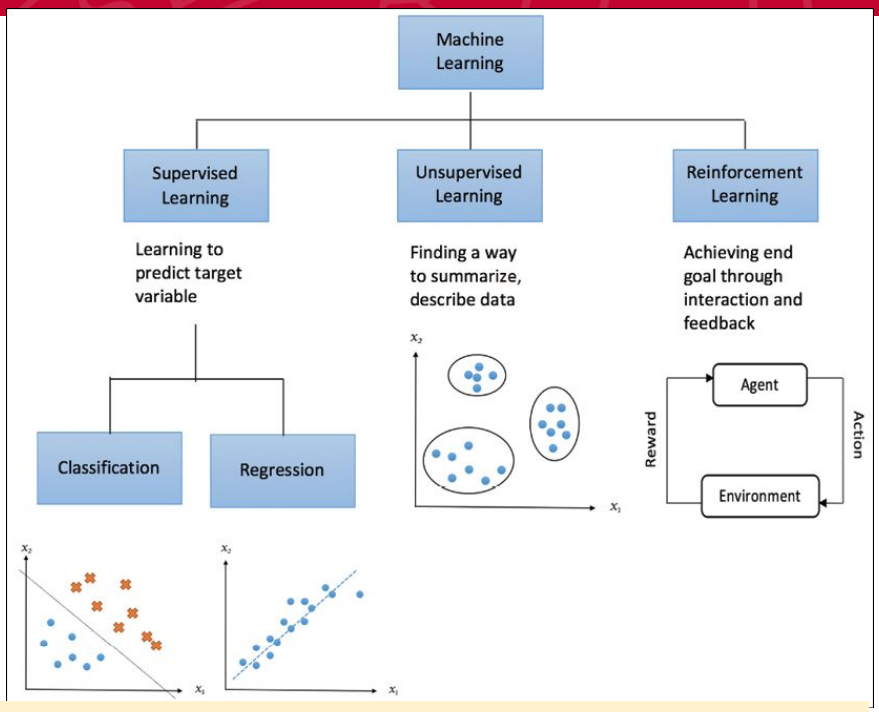


Figure 1.4: Types of machine learning tasks, del libro Yuxi (Hayden) Liu: *Python Machine Learning By Example, Third Edition*. Packt Publishing, 2020

Aprendizaje supervisado

- Si tenemos esta serie:

5 → 7

12 → 14

3 → 5

123 → 125

- ¿Cuál será la solución para los siguientes casos ?

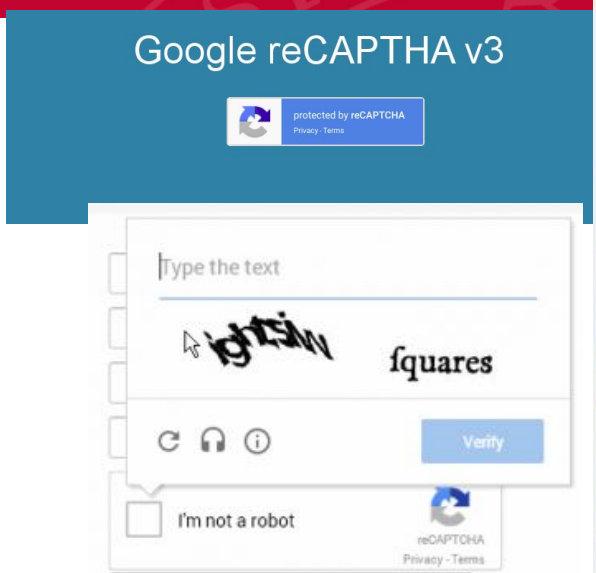
26 → ?

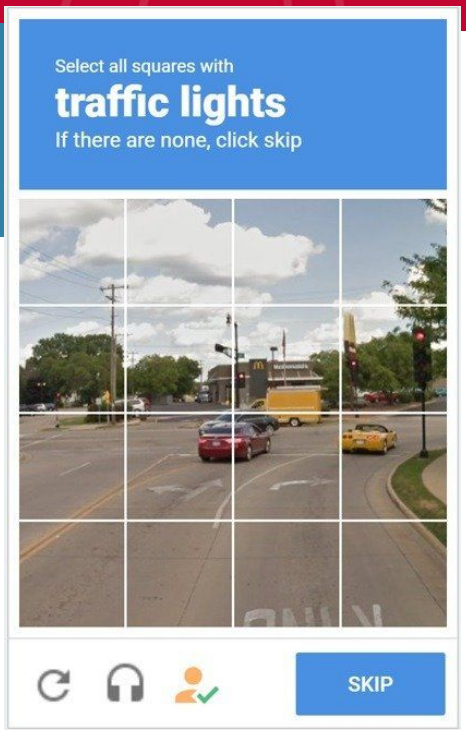
2 → ?

7 → ?

⇒ Mediante observación generalizamos el conocimiento

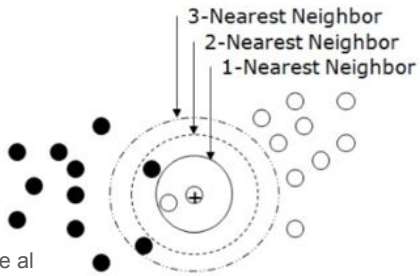
Google Recaptcha





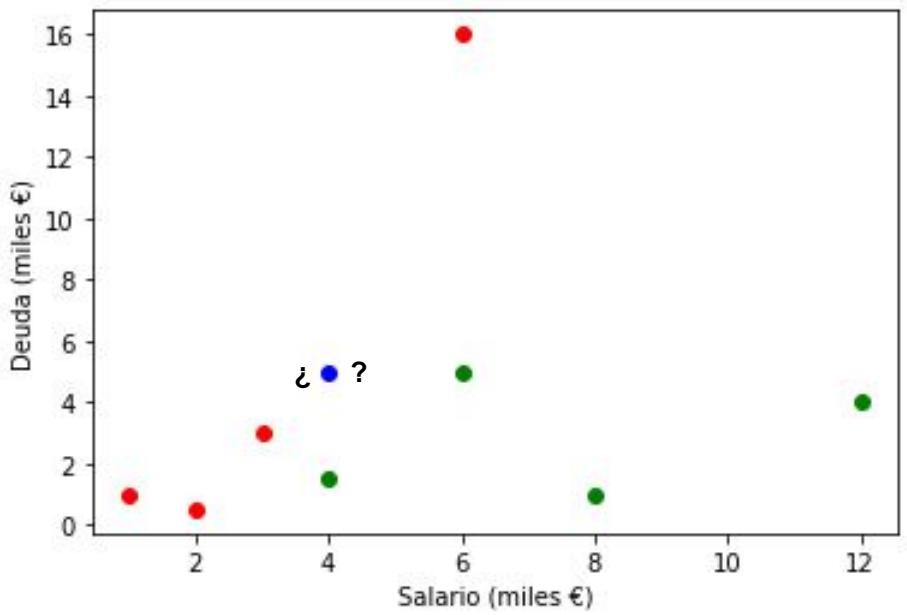
Los k-vecinos más cercanos (k-NN)

- Los **k-vecinos más cercanos** (o **k-NN** por *k-nearest neighbor*) es uno de los algoritmos más sencillos
  - Es **supervisado**: parte de un conjunto de datos (ejemplos) de entrenamiento, esto es, que para cada ejemplo se conoce su resultado
  - Se dice **basado en instancias** (ejemplos) porque no aprende explícitamente un modelo (como ocurre con la regresión logística), sino que utiliza los ejemplos de entrenamiento para la predicción
- Considera la distancia de un punto a los k ejemplos (instancias) conocidos más cercanos para determinar a qué clase pertenece o el valor que le corresponde
  - Utiliza una medida de distancia: euclídea, L1, L, Mahalanobis, ...
  - Devuelve como valor solución la media de los valores solución de los k más cercanos
    - Puede ser una media aritmética estándar o una media aritmética ponderada, donde se dé más peso a un vecino según lo cerca que se encuentre al valor para el que queremos determinar la solución



Ejemplo k-NN

- Determinar si dar un préstamo a un cliente



Regresión lineal

- Modelo matemático que permite aproximar la relación de dependencia entre una variable dependiente *Y* y variables independientes X_i

$$y = w_0 + w_1 x_1 + w_2 x_2 + \ldots$$

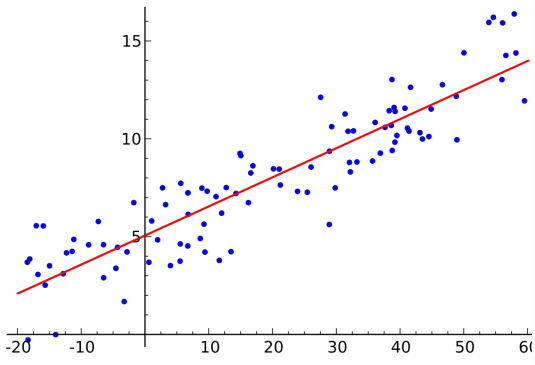
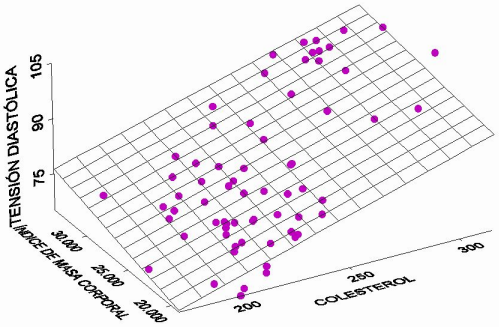


Figura 1.- Plano de regresión para la tensión arterial diastólica ajustando por colesterol e índice de masa corporal.



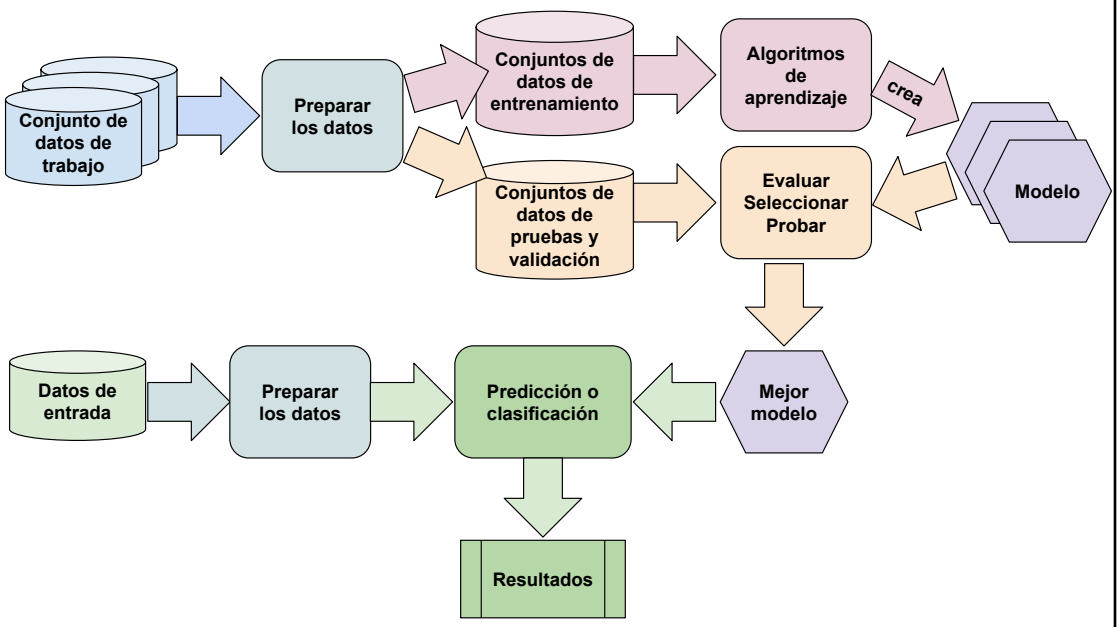
DASI

Conceptos básicos de aprendizaje automático

Generalización

- El aprendizaje automático es un **proceso inductivo**, de **generalización**
  - Deriva un modelo general a partir de ejemplos específicos
- Los datos se suelen representar como una matriz de números
  - Cada fila representa un **ejemplo** que se puede usar para entrenamiento, validación o prueba
  - Cada columna representa una variable, que se denomina **característica** (***feature***)
    - En aprendizaje supervisado una de las variables no es una característica, sino la **etiqueta** que corresponde a ese ejemplo
- Hay varias técnicas a considerar para obtener buenos modelos, que generalicen bien:
  - Evitar sobreajuste (*overfitting*) y falta de ajuste (*underfitting*)
  - Preprocesar los datos
  - Combinar modelos

Desarrollo de aplicaciones de Machine Learning



Conjuntos de datos

- Datos de **entrenamiento** (*training data set*)
  - Para configurar los parámetros del modelo
  - En torno al 80% de los datos disponibles (al menos el 60%)
- Datos de **validación** (*validation data set*)
  - Para el ajuste fino de los hiper-parámetros del modelo (los que controlan el proceso de aprendizaje)
  - No se usan para entrenamiento, sino para una evaluación objetiva del sesgo (bias) y varianza
  - Suele ser una parte de los datos de entrenamiento, aproximadamente 10-15% de los datos disponibles
- Datos de **prueba** (*test data set*)
  - Independiente del conjunto de datos de entrenamiento pero con una distribución de probabilidad de clases similar
  - Sirve de benchmark para evaluar el modelo final, esto es, una vez que su entrenamiento ha concluido
  - Aproximadamente el 20% de los datos disponibles

Errores de predicción

- **Sesgo (*bias*)**
  - Diferencia entre la predicción media ($\hat{y}$) y el valor real (y)

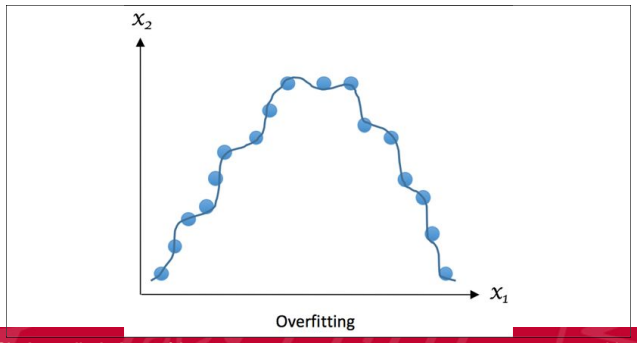
    $Bias[\hat{y}] = E[\hat{y}-y]$
- **Varianza**
  - Medida de la dispersión de la predicción (esto es, cuánto diferirá la predicción para una variable aleatoria)
  - Se calcula como la esperanza del cuadrado de la desviación de dicha variable respecto a su media

    $Varianza = E[\hat{y}^2] - E[\hat{y}]^2$
- **Error irreducible** (*ruido*)
  - No se puede corregir, independientemente del algoritmo
  - Variables desconocidas, conjunto de características incompleto, problema mal definido

Error (Modelo) = Sesgo^2 + Varianza + Error irreducible

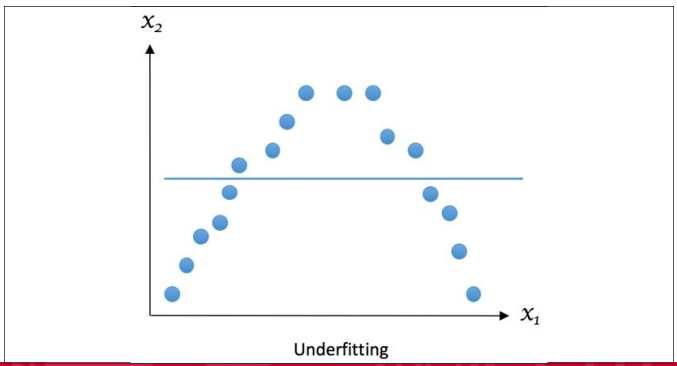
Sobreajuste (*Overfitting*)

- **Sobreajuste (Overfitting)**: el modelo se ajusta demasiado bien a las observaciones existentes pero falla al predecir con nuevos casos
  - Es como si se memorizaran todos los casos de entrenamiento y no se supiera qué responder ante nuevos casos
  - Se produce al extraer demasiada información del conjunto de datos de entrenamiento (por **sobreentrenamiento** del algoritmo de aprendizaje)
    - Demasiados parámetros para un conjunto de datos de entrenamiento pequeño
  - Cuanto más sobreajuste el éxito con los casos de entrenamiento sigue incrementándose mientras que con muestras nuevas va empeorando
- Cuando hay sobreajuste
  - **Baja el sesgo** (***bias***)
  - **Aumenta la varianza**

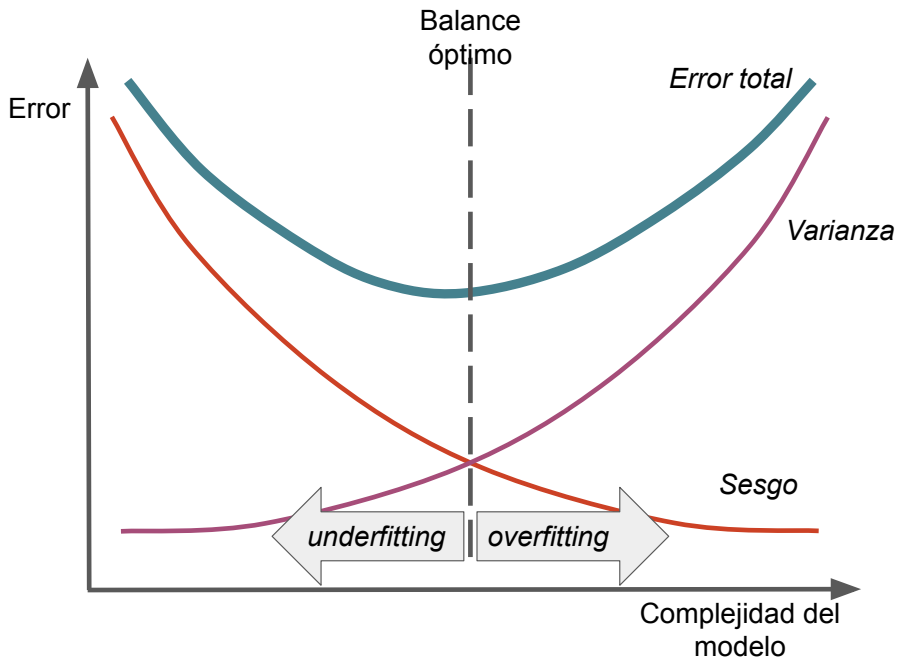


Falta de ajuste (underfitting)

- **Falta de ajuste (underfitting)**: el modelo no funciona bien ni con el conjunto de entrenamiento ni con el de prueba
  - Puede que no haya suficientes datos para entrenar el modelo o que el modelo a entrenar no sea el adecuado
- Se puede observar
  - **Sesgo** (***bias***) **alto**
  - **Varianza baja** (porque la eficiencia del modelo es mala tanto para el conjunto de entrenamiento como para el de prueba)

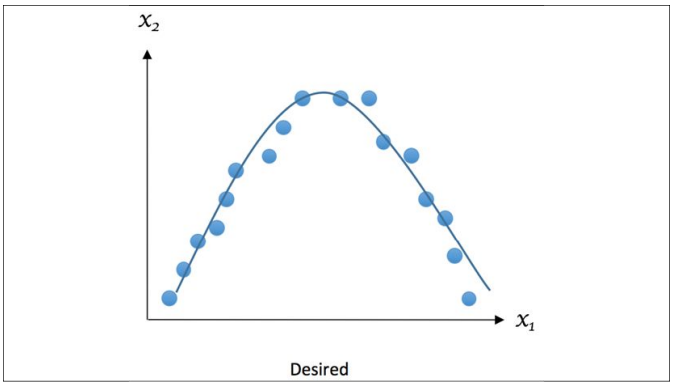


Objetivo: balance entre sesgo y varianza



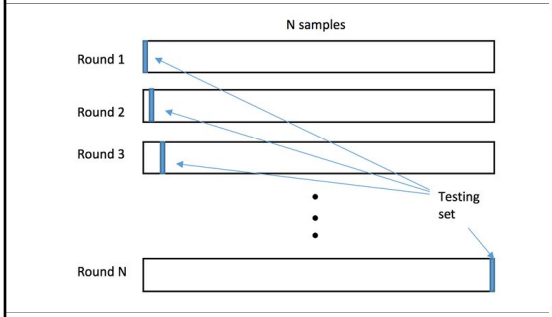
Compromiso sesgo-varianza

- Hay que evitar que tanto el sesgo como la varianza sean altos
  - Aunque normalmente bajar el sesgo aumenta la varianza
- Técnicas
  - Regularización
  - Validación cruzada
  - Reducción de características (*features*)



Validación cruzada (*cross-validation*)

- Los datos van usándose unas veces para entrenamiento y otras para pruebas/validación
  - Exhaustivo (p.ej., ***Leave-One-Out-Cross-Validation, LOOCV***): cada elemento está una vez en el conjunto de validación
  - No exhaustivo (p.ej. ***k-fold cross-validation***): se divide el conjunto de datos en conjuntos de k elementos y cada uno de estos se utiliza en una ronda como validación



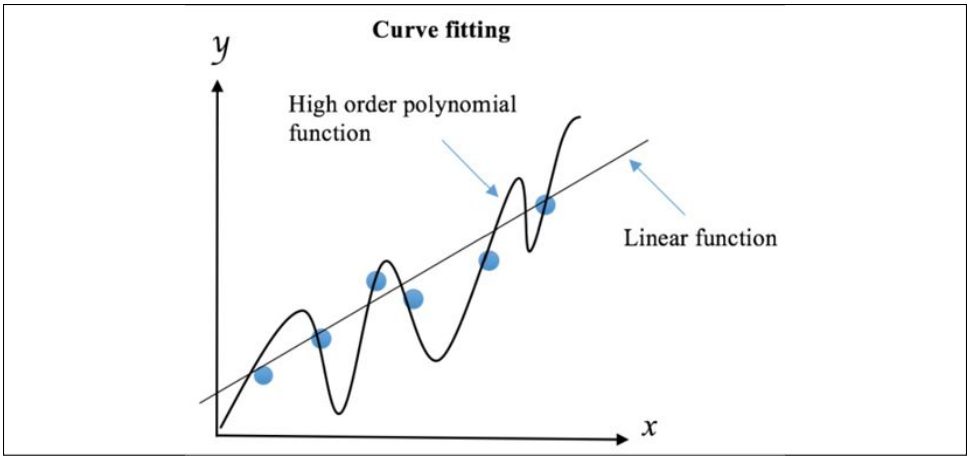
Round	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
1	**Testing**	Training	Training	Training	Training
2	Training	**Testing**	Training	Training	Training
3	Training	Training	**Testing**	Training	Training
4	Training	Training	Training	**Testing**	Training
5	Training	Training	Training	Training	**Testing**

Validación cruzada (*cross-validation*)

- Método ***Holdout***: Se generan conjuntos de validación de manera aleatoria varias veces
  - Algunos elementos pueden repetirse o no estar nunca en los conjuntos de prueba
- Validación cruzada anidada (**nested cross-validation**): combina varias validaciones cruzadas, en dos fases:
  - Validación cruzada **interna**: Sirve para encontrar el modelo más ajustado, normalmente usando k-fold cross-validation
  - Validación cruzada **externa**: Sirve para evaluar la eficiencia (performance) y análisis estadístico

Regularización

- Si el modelo es demasiado complejo aumentará el sobreajuste



⇒ La regularización añade parámetros extra a la función de error que se quiere minimizar para penalizar los modelos complejos

Regularización

- Considérese un perro robot guardián que tiene que identificar extraños
  - Se le entrena con el siguiente conjunto de datos:

Hombre	Joven	Alto	Con gafas	En gris	**Amigo**
Mujer	Mediana	Normal	Sin gafas	En negro	**Extraño**
Hombre	Joven	Bajo	Con gafas	En blanco	**Amigo**
Hombre	Mayor	Bajo	Sin gafas	En negro	**Extraño**
Mujer	Joven	Normal	Con gafas	En blanco	**Amigo**
Hombre	Joven	Bajo	Sin gafas	En rojo	**Amigo**

  - Un conjunto de reglas que podría aprender es:

```
Cualquier mujer de mediana edad, normal, sin gafas y en negro, es un extraño
Cualquier hombre mayor, bajo, sin gafas, en negro, es un extraño
Cualquier otro, es amigo
```

→ Se adecúa perfectamente al conjunto de entrenamiento, pero ¿generaliza bien para otros casos?

Regularización

- Es un modelo de reglas demasiado complejo
  - Tal vez fuera más apropiado algo más sencillo, limitando los aspectos aprendidos:

    ```
    Cualquiera sin gafas, en negro, es un extraño
    ```

- Se puede parar el proceso de aprendizaje como una forma de regularización
  - **Limitando el tiempo de aprendizaje**
  - Poniendo algún **criterio interno de parada**

    ⇒ **Parada temprana (*early stopping*)**

- Cuidado: Una regularización demasiado exagerada puede ocasionar falta de ajuste (*underfitting*)

Selección de características (*feature selection*)

- El número de características se corresponde con la dimensionalidad de los datos
  - Y el aprendizaje automático depende del número de dimensiones vs. el número de ejemplos
  - Tratar con datos de gran dimensionalidad es muy costoso computacionalmente
- No todas las características de los datos son útiles y pueden añadir aleatoriedad a los resultados, o algunas pueden ser redundantes o irrelevantes
- La **selección de características** es el proceso de elección de características significativas para generar un mejor modelo
- Otro mecanismo para reducir la dimensionalidad de los datos es la **proyección de características** (*feature projection*)

Combinación de modelos

- Se pueden procesar los datos con varios modelos y obtener un resultado como combinación de los de todos los modelos
  - **Voto**: se toma el resultado de la mayoría
  - **Media**: se hace una media que puede ser ponderada
  - ***Bagging*** (*Bootstrap aggregating*): Aplica bootstrapping que consiste en:
    - Generar varios conjuntos de entrenamiento mediante muestreo con sustitución
    - Para cada conjunto entrena un modelo nuevo
    - Combina los resultados con la media o voto de la mayoría

    ⇒ Se reduce el overfitting al combinar los modelos
  - ***Boosting***: los modelos se entrenan en secuencia en vez de en paralelo (*bagging*), todos con el mismo conjunto de datos, pero cada ejemplo de datos con un factor de peso que depende del éxito en el modelo anterior
    - En general, los pesos de los ejemplos mal predichos se incrementan más para destacar su dificultad de predicción
  - **Apilamiento (*Stacking*)**: toma los valores de salida de los modelos de aprendizaje y los utiliza como valores de entrada para otro algoritmo

Ejemplo de equilibrio entre sesgo y varianza

Ver en Google Colab:
https://colab.research.google.com/drive/1H89iG5GoV6N6E24DO9zmB4ZPK_2bKlGf?usp=sharing

DASI

Preprocesado de datos

Preprocesado de datos

- Los algoritmos de aprendizaje automático necesitan los datos preparados con cierto formato y características
  - Convertir datos crudos (raw data) en el formato apropiado
  - Limpiar el conjunto de datos
- Operaciones comunes de preprocesado de datos
  - Limpieza y filtrado de datos erróneos o incompletos
  - Re-escalado - Cambia los valores para que estén en un rango
  - Estandarización - Resta la media y divide por la desviación estándar
  - Normalización - Escala los valores en un rango (0..1)
  - Binarización - Deja los valores como 0 o 1 (p.ej. llovió o no)
- Varias utilidades:
  - Funciones de Pandas para limpiar datos erróneos o filtrado
  - La librería **scikit-learn** es útil para preparar los datos
    ⇒ https://scikit-learn.org

Preprocesado de datos

- Ejemplos de problemas que se pueden encontrar con los datos
  - **Errores humanos**: que se haya puesto 200 en vez de 2000, typos, varias versiones de la misma entidad (CAM, cam, Comunidad de Madrid)
  - **Valores inesperados**: en vez de un número en alguna celda hay un carácter textual y eso hace que toda la columna se considere strings
  - **Información incompleta**: en una encuesta preguntas sin respuesta, o por un error del sistema informático no se han registrado algunos datos
  - **Resolución**: se han recogido datos cada segundo pero para el análisis se requieren por hora
  - **Relevancia de los campos**: Algunos de los datos proporcionados no son necesarios para el análisis y hay que limpiarlos
  - **Formato de los datos**: pueden estar registrado en un formato que no es apropiado para la herramienta de análisis y habrá que adaptarlos

Fichero de ejemplo: datos.csv

```
Duración,Fecha,Pulsaciones,Máx pulsaciones,Calorías
60,2020/12/01,110,130,409.1
60,2020/12/02,117,145,479.0
60,2020/12/03,103,135,340.0
45,2020/12/04,109,175,282.4
45,2020/12/05,117,148,406.0
60,2020/12/06,102,127,300.0
60,2020/12/07,110,136,374.0
450,2020/12/08,104,134,253.3
30,2020/12/09,109,133,195.1
60,2020/12/10,98,124,269.0
60,2020/12/11,103,147,329.3
60,2020/12/12,100,120,250.7
60,x2020/12/12,100,120,250.7
60,2020/12/13,106,128,345.3
60,2020/12/14,104,132,379.3
60,malafecha,98,123,275.0
60,2020-12/16,98,120,215.2
60,2020/12/17,100,120,300.0
45,2020/12/18,90,112,NaN
60,2020/12/19,103,123,323.0
45,2020/12/20,97,125,243.0
60,2020/12/21,108,131,364.2
45,NaT,100,119,282.0
60,,130,101,300.0
45,2020/12/24,105,132,246.0
60,2020/12/25,102,126,334.5
60,2020/12/26,100,120,250.0
60,2020/12/27,92,118,241.0
60,2020/12/28,103,132,NaN
60,2020/12/29,100,132,280.0
60,2020/12/30,102,129,380.3
60,2020/12/31,92,11,
```

Campos separados por coma

Duplicados

Fecha con mal formato

Algunos campos no tienen un valor numérico

Falta la fecha es NaT (not a time)

Habrá que tratar todos estos errores en los datos

Otros están vacíos

Información del dataframe

- Si se crea un dataframe a partir del fichero, se puede ver una descripción con info():

```
df=pd.read_csv("datos.csv")
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Duración         32 non-null     int64
 1   Fecha            32 non-null     object
 2   Pulsaciones      32 non-null     int64
 3   Máx pulsaciones  32 non-null     int64
 4   Calorías         29 non-null     float64
dtypes: float64(1), int64(3), object(1)
memory usage: 1.4+ KB
```

Hay 32 filas, que están indexadas como un rango de enteros de 0 a 31

La mayoría de las columnas tienen los 32 elementos con valores aparentemente válidos

Indica que solo hay 29 valores no nulos (porque hay 3 NaN, uno de ellos del campo vacío)

Celdas vacías

- Las celdas vacías pueden ocasionar fallos en el análisis de los datos
  - Se consideran celdas vacías las que tienen valor `None` (nada) o `numpy.NaN`
- La solución más fácil es **eliminar** las filas correspondientes
  - Normalmente los conjuntos de datos son muy amplios, así que no afectará al resultado del análisis de manera significativa
- Otra solución es **rellenarlas** con algún valor
  - Media, mediana, moda, de la columna
  - Algún otro valor calculado según la fórmula que se considere
- Hay varios métodos de Pandas para tratar celdas vacías:
  - **`isna()`** (o **`isnull()`**)- indica con True o False las celdas llenas o vacías
  - **`fillna()`** - reemplaza los valores que faltan
  - **`dropna()`** - elimina la fila (o columna) con valores que faltan

Tratamiento de celdas vacías

```
df2=df.dropna()
```

Devuelve un nuevo dataframe eliminando las filas correspondientes, pero no cambia df

```
df.dropna(inplace = True)
```

No devuelve nada, y cambia el df (al poner **inplace=True**)

Cambia en el dataframe todas las celdas sin valor correcto o nulo por 0

```
df.fillna(0, inplace = True)
```

Similar pero solo cambia valores en la columna "Calorias"

```
df["Calorías"].fillna(0, inplace = True)
```

```
media = df["Calorías"].mean()
df["Calorías"].fillna(media, inplace = True)
```

El valor que asigna es la media de la columna

Podría hacerse también con la mediana (**`median()`**), la moda (**`mode()`**), etc.

Celdas con formato erróneo

- La columna "Fecha" tiene alguna que no está bien pero para detectarla antes habría que indicar que los datos de esta columna son del tipo Date
  - **`to_datetime()`** - convierte los objetos indicados al tipo datetime64

En qué columna se pone el resultado (una nueva o en este caso la misma)

Si se producen errores al parsear la fecha no da excepción y se genera NaT

```
df['Fecha'] = pd.to_datetime(df['Fecha'], errors = 'coerce')
```

```
df.dropna(subset=['Fecha'], inplace = True)
```

Elimina todas las filas donde haya un NaT

Celdas con valores erróneos

- A veces hay errores en los valores (demasiado grandes o pequeños, por ejemplo)
  - En el ejemplo hay una celda con duración 450 minutos, dispar con todo el resto

  ⇒ Se puede recorrer todo el dataframe con un for para identificar y tratar esos valores: cambiar el valor o eliminar la fila

```
duracion_max = 120

for i in df.index:
    if df.loc[i, "Duración"] > duracion_max:
        df.loc[i, "Duración"] = duracion_max

        df.drop(i, inplace = True)
```

Una opción es cambiar el valor

Otra opción es eliminar la fila

Eliminar duplicados

- A veces puede haber filas duplicadas que conviene eliminar
  - `duplicated()` permite identificar filas repetidas de un dataframe y devuelve una serie con booleanos indicando si una fila ya existía igual
  - Para eliminar las duplicadas: `df.drop_duplicates()`

```
print(df.duplicated())

filasduplicadas=df[df.duplicated()]

df.drop_duplicates(inplace = True)
```

Imprime True o False por cada fila si está duplicada o no

Devuelve un dataframe con todas las filas duplicadas

Elimina todas las filas duplicadas en el dataframe

Ejercicio

- Crea un Dataframe a partir del fichero de datos del covid mundiales de https://covid.ourworldindata.org/data/owid-covid-data.csv
  - A partir de este DataFrame crea otro que tenga solo los datos de España
  - Limpiar el DataFrame resultante haciendo varios ajustes:
    - En las columnas *new_cases_smoothed, total_deaths, new_deaths* y *new_deaths_smoothed,* poner a 0 las celdas vacías
    - Poner como objetos tipo date las celdas de la columna date
    - Sacar un listado de las filas donde el número de casos (*new_cases*) sea negativo
    - Comprobar si hay filas duplicadas y si las hubiera, eliminarlas
    - Guardar el resultado en un fichero excel
  - Guardar el resultado en un fichero Excel y comprobar que está bien

Reescalado de datos

- Es común que los atributos de los datos vengan con diferentes escalas y sea necesario pasarlos a una misma escala

  ⇒ Si se reescalan a un rango de 0 a 1 se denomina **normalización**

- La clase **`MinMaxScaler`** de scikit-learn implementa un escalador
  - Transforma los valores de las características (features) para que estén en un rango determinado
  - Para utilizarlo:
    - Importar la clase

      ```
      from sklearn.preprocessing import MinMaxScaler
      ```

    - Crear un objeto escalador

      ```
      scaler = MinMaxScaler()
      ```

    - Y aplicar la transformación a los datos

      ```
      datos_escalados = scaler.fit_transform(datos.iloc[:, :x])
      ```

Reescalado de datos - MinMaxScaler

```
import pandas, scipy, numpy
from sklearn.preprocessing import MinMaxScaler

df=pandas.read_csv(
'http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/
winequality-red.csv ', sep=';')
array=df.values
# Separa los datos en componentes de entrada y salida
x=array[:,0:8]
y=array[:,8]
scaler=MinMaxScaler(feature_range=(0,1))
rescaledX=scaler.fit_transform(x)
numpy.set_printoptions(precision=3) #Setting precision for the output
rescaledX[0:5,:]
```


Codificación de etiquetas

Etiqueta	Código
rojo	1
amarillo	2
verde	3
azul	4
otro	5

Puede malinterpretarse un cierto orden en los colores

Etiqueta	esRojo	esAmar	esVerde	esAzul
rojo	1	0	0	0
amarillo	0	1	0	0
verde	0	0	1	0
azul	0	0	0	1
otro	0	0	0	0

Esquema ***one-of-K*** o ***one-hot encoding***:
Se utilizan variables ficticias, cuyo valor será binario.
Si todas toman el valor 0 entonces será otro

Este tipo de matriz con tantos 0 es una matriz dispersa, que puede manejarse con cierta eficiencia (p.ej. el módulo `sparse` de la librería `scipy`)

DASI

Bibliografía

Bibliografía

- Sridhar Alla & Suman Kalyan Adari: *Beginning MLOps with MLFlow. Deploy Models in AWS SageMaker, Google Cloud, and Microsoft Azure*. Apress, 2021.
- Stuart Russell, Peter Norvig: *Artificial Intelligence: A Modern Approach, 3rd edition*. Prentice Hall, 2016.Teik Toe Teoh & Zheng Rong: Artificial Intelligence with Python. Springer Nature, 2022.
- Yuxi (Hayden) Liu: *Python Machine Learning By Example, Third Edition*. Packt Publishing, 2020.
- Documentación en línea de las distintas herramientas