Módulo 3 - Ciencia de Datos y Machine Learning.

3.2 Aprendizaje Supervisado.

Autor:

Por Prof. Alberto Fernández Hilario

Profesor Catedrático de Universidad de Granada. Instituto Andaluz Interuniversitario en Data Science and Computational Intelligence (DasCI)

Breves Instrucciones

Recordatorio: Introducción a NoteBook

El cuaderno de *Jupyter* (Python) es un enfoque que combina bloques de texto (como éste) junto con bloques o celdas de código. La gran ventaja de este tipo de celdas, es su interactividad, ya que pueden ser ejecutadas para comprobar los resultados directamente sobre las mismas.

Muy importante: el orden de las instrucciones (bloques de código) es fundamental, por lo que cada celda de este cuaderno debe ser ejecutada secuencialmente. En caso de omitir alguna, puede que el programa lance un error (se mostrará un bloque salida con un mensaje en inglés de color rojo), así que se deberá comenzar desde el principio en caso de duda. Para hacer este paso más sencillo, se puede acceder al menú "Entorno de Ejecución" y pulsar sobre "Ejecutar anteriores".

¡Ánimo!

Haga clic en el botón "play" en la parte izquierda de cada celda de código. Las líneas que comienzan con un hashtag (#) son comentarios y no afectan a la ejecución del programa.

También puede pinchar sobre cada celda y hacer "*ctrl+enter*" (*cmd+enter* en Mac).

Cuando se ejecute el primero de los bloques, aparecerá el siguiente mensaje:

"*Advertencia: Este cuaderno no lo ha creado Google.*

El creador de este cuaderno es \<autor>@go.ugr.es. Puede que solicite acceso a tus datos almacenados en Google o que lea datos y credenciales de otras sesiones. Revisa el código fuente antes de ejecutar este cuaderno. Si tienes alguna pregunta, ponte en contacto con el creador de este cuaderno enviando un correo electrónico a \<autor>@go.ugr.es."

No se preocupe, deberá confiar en el contenido del cuaderno (Notebook) y pulsar en "*Ejecutar de todos modos*". Todo el código se ejecuta en un servidor de cálculo externo y no afectará en absoluto a su equipo informático. No se pedirá ningún tipo de información o credencial, y por tanto podrá seguir con el curso de forma segura.

Cada vez que ejecute un bloque, verá la salida justo debajo del mismo. La información suele ser siempre la relativa a la última instrucción, junto con todos los `print()` (orden para imprimir) que haya en el código.

ÍNDICE

En este *notebook*:

1. Se retomará el concepto de Aprendizaje Supervisado y sus fundamentos.
2. Se realizará una discusión sobre cuáles son las características principales que influyen en dicho aprendizaje.
3. Se introducirá el concepto de Inteligencia Artificial confiable y su influencia en las decisiones a tomar en la recolección de datos, adaptación de los modelos, y auditoría de los resultados.
4. Se explicará la importancia de la validación de los modelos de aprendizaje.
5. Se presentarán distintas alternativas de validación.

Contenidos:

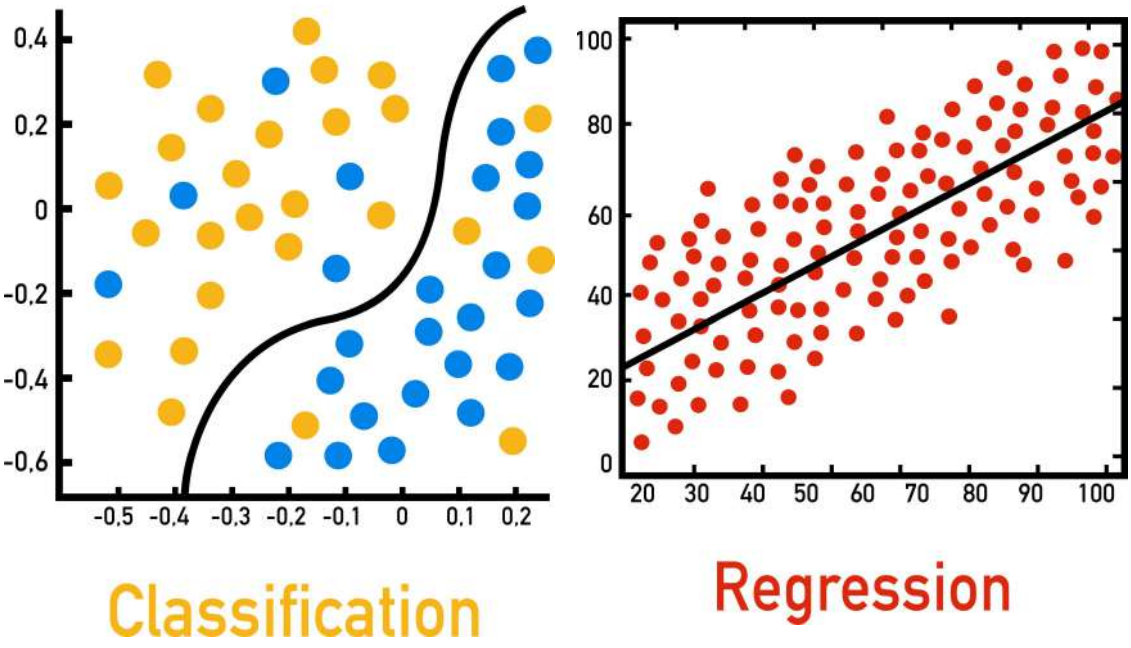
1. ¿QUÉ ES EL APRENDIZAJE SUPERVISADO?

En módulos anteriores del curso, se han destacado algunos ejemplos de interés sobre el uso de Machine Learning en el campo de la bioinformática. En concreto, dentro de la Cápsula 2 del Módulo 1 (*La Bioinformática. Aplicaciones en Bio-Ciencias y Bio-Salud*), se enumeraron el desarrollo y descubrimiento de fármacos, la microbiología, minería de textos biomédicos, medicina de precisión o personalizada, entre otros. Siendo más específicos, un uso directo del Machine Learning en estos campos sería como herramienta de diagnóstico, para determinar la categoría a la que pertenece un paciente (sano o enfermo). Otra opción válida sería ajustar el nivel de dosis concreto de un medicamento, en este caso calculando un valor numérico.

En los dos ejemplos anteriores, el objetivo es realizar una predicción del valor de una variable de salida, como el tipo de paciente, o la dosis del medicamento. Cuando el ser humano realiza esta tarea, la realiza en base a su conocimiento basado en la experiencia con otros casos similares (pacientes y medicamentos). En el caso de Machine Learning, esta experiencia se recopila a partir de instancias en un conjunto de datos, como ya se indicó en la primera cápsula de este Módulo.

Tanto el diagnóstico de pacientes, como la estimación de la dosis de un medicamento, se encuentran dentro de lo que se denomina como "*Aprendizaje supervisado*". De acuerdo al tipo de variable de salida del problema, las dos tareas fundamentales dentro del aprendizaje supervisado son la **clasificación** y la **regresión**.

Se denomina clasificación cuando el objetivo es determinar la categoría de una instancia dentro de un grupo de valores fijos (el diagnóstico de pacientes). Por su parte, la regresión busca, a grandes rasgos, crear una función matemática de interpolación para una variable de tipo real (la estimación de dosis del medicamento).



En ambos casos, se utilizarán las variables de entrada que definen el problema o caso de estudio. En concreto, los algoritmos de Machine Learning suelen buscar correlaciones altas entre las variables de entrada y las de salida para así construir un modelo de buena calidad.

2. CARACTERÍSTICAS DE LOS DATOS QUE INFLUYEN EN EL APRENDIZAJE

Dentro de las tareas de aprendizaje supervisado, es decir, clasificación y regresión, el algoritmo de aprendizaje busca ajustarse lo mejor posible a los datos durante el entrenamiento. Para ello, se utilizan las variables de entrada que describen a cada instancia del problema.

En este sentido, un tema importante a discutir es si resulta posible determinar la cantidad de datos que es óptima para realizar un correcto aprendizaje, así como la relación entre el número de instancias y el número de variables que las representan. Otra cuestión muy

relevante es comprender los detalles asociados a las variables de entrada en sí mismas. Por último, se debe poner énfasis en la necesidad de datos de calidad y el sesgo en los mismos.

2.1 Discusión respecto al número de instancias

Con respecto al volumen de información necesario para la tarea de aprendizaje supervisado, no existe una respuesta general que sea válida para todos los casos de estudio o problemas.

Desde el punto de vista estadístico, la muestra (el conjunto de datos) será más representativa cuanto mayor sea su número. De este modo, a mayor volumen de información disponible, más casos diferentes se es capaz de cubrir, y mejor se adaptará el modelo al caso de estudio real.

Sí que es posible extraer conocimiento a partir de unos pocos datos, 50 ó 100 por dar un número aproximado. Sin embargo, existe una alta probabilidad que el modelo generado sea demasiado específico, y por tanto no sea útil para su posterior aplicación. El número correcto dependerá de la dificultad del problema a resolver, si bien una regla genérica es que el número de instancias necesario para un correcto aprendizaje debe ser al menos 10 veces el número de parámetros utilizados para configurar el algoritmo.

Finalmente, esta la cuestión sobre si existe una ratio específica entre el número de instancias y variables que sea apropiada para realizar un aprendizaje correcto. Desafortunadamente, tampoco hay una respuesta exacta, si bien se puede reutilizar la regla genérica anterior para indicar que, en este caso, el número de instancias debe ser al menos 10 veces el número de variables de entrada utilizadas.

2.2 Discusión sobre las variables de entrada

Tal como se indicó anteriormente, los algoritmos de aprendizaje automático suelen buscar correlaciones altas entre las variables de entrada y las de salida.

La mayor parte de algoritmos de Machine Learning tienen cierta preferencia sobre las variables de entrada de tipo numérico (valores reales como el nivel de expresión genética, o el ph de un producto), si bien se pueden utilizar también variables de tipo nominal (categorías como color o género).

Lo único importante en este caso es tener cuidado con incluir variables que puedan desvirtuar el conocimiento que se desea extraer. Por ejemplo, podría existir una relación fuerte entre la edad de una persona y el tiempo de supervivencia a una enfermedad concreta, pero nuestro objetivo es que esa asociación la encontremos directamente en los valores genéticos. Este tipo de variables se conocen como "**confounding**" y causan asociaciones de tipo espúreas que deben ser identificadas y evitadas a toda costa, especialmente cuando se trabaja en el área de la bioinformática.

Otro tema relevante con respecto a las variables se define como "la maldición de dimensionalidad". El término dimensionalidad se refiere la cardinalidad (número de elementos) del conjunto de variables. En los problemas de tipo bioinformático, el número de variables utilizadas para el estudio suele ser de miles o incluso decenas de miles. Este hecho

dificulta muchísimo la tarea de aprendizaje, ya que se disminuye la capacidad del algoritmo de encontrar una correlación óptima entre las entradas y la salida.

Por este motivo, resulta imprescindible dedicar un esfuerzo considerable en la fase de preparación de los datos, seleccionando exclusivamente aquellas variables que sean más importantes para el estudio. En efecto, se suele comprobar que un modelo construido sobre un subconjunto de variables predictivas de calidad siempre es mejor que aquél generado utilizando todas las variables originales.

3. INTELIGENCIA ARTIFICIAL CONFIABLE: GARANTIZANDO ÉTICA EN LOS MODELOS

En un mundo impulsado por la Inteligencia Artificial (IA) y la Ciencia de Datos, la confiabilidad y ética son fundamentales. Recuerda que cuando hablamos de IA nos estamos refiriendo a los sistemas que utilizan Machine Learning, pues actualmente ambos términos están íntimamente conectados.

En esta sección, exploraremos la creciente importancia de garantizar la transparencia, equidad y seguridad en el uso de la IA, especialmente en contextos biológicos.

3.1 Inteligencia Artificial Explicable y Transparencia de los Modelos

Garantizar que los sistemas de IA y/o modelos de Machine Learning sean transparentes y sus decisiones sean explicables es esencial. En biología, imagina un modelo que ayuda a diagnosticar enfermedades basado en datos genómicos. Es crucial que los biólogos puedan entender cómo se llega a un diagnóstico. La transparencia de los modelos y su explicabilidad aseguran que las decisiones de la IA sean confiables y éticas.

Para garantizar lo anterior, algunas de las estrategias más comunes serían las siguientes:

1. **Interpretabilidad del Modelo**: Esto implica elegir modelos de Machine Learning que sean naturalmente más fáciles de entender. En el Módulo 5, analizaremos cuáles son las propiedades que determinan cuándo un modelo tiende a ser más interpretable, lo cual es especialmente debido a la posibilidad de trazar su lógica de toma de decisiones.

2. **Visualización de Datos y Modelos**: Las representaciones visuales pueden ayudar a los usuarios a comprender mejor cómo funciona un modelo. Gráficos, diagramas y representaciones visuales de datos y resultados modelados pueden hacer que la información sea más accesible.

3. **Técnicas de Explicabilidad**: Se han desarrollado métodos específicos para explicar modelos más complejos, es decir, aquellos que no son interpretables per se. Uno de los enfoques es el uso de gráficos de importancia de características, que muestran qué variables son más influyentes en las predicciones del modelo.

4. **Documentación y Metadatos**: Mantener registros detallados y documentación de modelos y datos es esencial para la transparencia. Esto incluye información sobre cómo se recopilaron los datos, cómo se entrenó el modelo y cómo se tomaron las decisiones.

3.2 Abordando el Sesgo y la Equidad (fairness) en los Datos

En el emocionante mundo de la Ciencia de Datos, existe una preocupación importante que va más allá de las matemáticas y las técnicas de modelado: el sesgo. El sesgo en los datos puede llevar a resultados injustos, inequitativos y éticamente problemáticos. En este contexto, el sesgo se refiere a la presencia de prejuicios en los datos que pueden afectar negativamente a ciertos grupos o resultados. Por ejemplo, si un modelo de Machine Learning utilizado en conservación de la biodiversidad se entrena principalmente con datos de una región geográfica, puede ignorar problemas en otras áreas.

Una idea fundamental que debemos comprender es que **los algoritmos en sí mismos no son inherentemente sesgados**. En cambio, es el sesgo que reside en los propios datos lo que puede introducir resultados no equitativos. Esto significa que, para crear sistemas de Inteligencia Artificial y Machine Learning justos y éticos, debemos prestar atención tanto al proceso de recopilación y preparación de datos como al diseño y entrenamiento de nuestros modelos.

Un aspecto clave en la comprensión del sesgo de datos es la noción de **variables protegidas**, que son características de los datos, como género, raza o edad, que están sujetas a posibles discriminaciones. Identificar estas variables protegidas es crucial para evaluar posibles fuentes de sesgo.

Un enfoque común es evaluar el sesgo es en función de grupos identificados por las anteriores variables protegidas. Así, en la actualidad es bastante común realizar una auditoría de modelos para examinar a fondo su comportamiento un modelo en relación con las denominadas como “métricas de sesgo”.

Imaginemos que estamos desarrollando un modelo de Machine Learning para predecir el riesgo de una persona de desarrollar una enfermedad cardíaca en función de sus datos genéticos y su historial de salud. En este contexto, dos variables clave podrían estar relacionadas con el sesgo: el género y la edad de los pacientes. El género es una variable binaria (hombre o mujer), y la edad se puede agrupar en diferentes categorías (por ejemplo, jóvenes, adultos y mayores). En este caso, podríamos centrarnos en la denominada como métrica de disparidad en la tasa de predicciones positivas, que se refiere a cuántos pacientes de cada grupo (género y edad) son etiquetados como "en riesgo" por el modelo. Por ejemplo, podríamos calcular la tasa de predicciones positivas para hombres y mujeres por separado y para cada grupo de edad. Si observamos diferencias significativas en estas tasas, podríamos estar frente a un sesgo en el modelo.

3.3 Seguridad, Robustez y Cumplimiento de la Normativa en la Inteligencia Artificial

En el ámbito de la Ciencia de Datos, es esencial comprender y aplicar conceptos relacionados con la seguridad, la robustez y la regulación en los algoritmos de Machine Learning. Estos conceptos se vuelven críticos para garantizar que los avances tecnológicos no solo sean efectivos, sino también seguros y éticos.

1. **Robustez de Algoritmos:** La robustez se refiere a la capacidad de un algoritmo de Machine Learning para mantener su rendimiento incluso en situaciones inusuales o cuando se enfrenta a datos ruidosos. En biología, esto se traduce en la capacidad de los modelos para analizar datos genéticos en los que pueden haber errores de secuenciación o variabilidad inesperada.

2. **Seguridad en Datos Biológicos:** Dada la sensibilidad de los datos biológicos, es fundamental garantizar su seguridad. Esto implica proteger la privacidad de los pacientes y asegurarse de que los datos genéticos o médicos no se utilicen de manera inapropiada. Por ejemplo, al compartir datos de pacientes para investigaciones, es crucial garantizar la anonimización y la protección de datos personales. Actualmente, el denominado como **Aprendizaje Federado** es un paradigma que permite entrenar modelos sin necesidad de compartir datos sensibles de diferentes fuentes.

3. **AI Act y Regulaciones:** El AI Act es un reglamento importante de la Unión Europea que busca regular la IA para garantizar su uso ético y seguro. Esto significa que cualquier investigador o desarrollador debe cumplir con regulaciones específicas para garantizar la seguridad y la ética en proyectos que contengan un apartado de IA. En concreto, se definen límites en el desarrollo de aplicaciones en función de su riesgo hacia las personas. No todos los proyectos de IA en biología presentan el mismo nivel de riesgo. Algunos, como el diagnóstico médico, pueden tener un alto impacto en la salud de las personas. Otros, como la identificación de patrones en secuencias genéticas, pueden tener un riesgo moderado. Entender y categorizar estos niveles de riesgo es crucial para aplicar la IA de manera responsable.

4. CASO DE ESTUDIO COMPLEMENTARIO: APRENDIZAJE DE CÁNCER DE MAMA MEDIANTE IMÁGENES

A lo largo de este curso, se trabajará sobre un problema de aprendizaje relativo al **melanoma cutáneo**. Sin embargo, en este apartado en concreto, por cuestión de simplicidad, vamos a realizar algunas pruebas iniciales utilizando como base un conjunto de datos relativamente sencillo, conocido como *breast cancer*.

Las variables de entrada de este conjunto se calcularon a partir de una imagen digitalizada de un aspirado de aguja fina (FNA) de una masa mamaria. Describen las características de los núcleos celulares presentes en la imagen en un espacio tridimensional.

Es un conjunto de datos pequeño y muy conocido en la literatura, así como disponible en muchos de las herramientas de Machine Learning, como *Scikit-Learn*.

Para utilizar este conjunto de datos, se procede a mostrar el código que permite guardar los datos en las variables de Python de una manera sencilla. Observe bien la estructura del bloque de código, que será descrita con detalle justo a continuación.

In [1]:
```
import pandas as pd
#Scikit-learn contiene su propio repositorio de datos
from sklearn.datasets import load_breast_cancer
```

```
data = load_breast_cancer()
#guardamos la entrada (características/variables) y salida (clase) en dos variables
X, y = data.data, data.target
#Transformamos las variables a de tipo numpy a tipo DataFrame (por comodidad de usc
X = pd.DataFrame(X,columns = data.feature_names)
y = pd.DataFrame(y,columns=["label"])

#Observamos las cinco primeras muestras
X.head()
```

Out[1]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809

5 rows × 30 columns

A partir del código anterior, se puede extraer la siguiente información:

- `X` e `y` son dos estructuras de datos tipo `dataframe` (marco de datos) que son almacenadas en tipo `Pandas` (*biblioteca muy usada en Python*). Un `dataframe` se entiende como un vector n-dimensional, resultando en general mucho más cómodo trabajar con este tipo de estructura de datos que con `numpy`.
- `X` contiene la matriz de datos de entrada e `y` es un vector unidimensional con la etiqueta de cada muestra en `X`. Por simplicidad, en este caso se ha optado por resolver un problema de clasificación (variable de salida de tipo categórica).
- En el aprendizaje de tipo supervisado, `X` e `y` se usarán para construir el modelo de Machine Learning y probarlo.

5. NECESIDAD DE VALIDAR LOS MODELOS DE MACHINE LEARNING

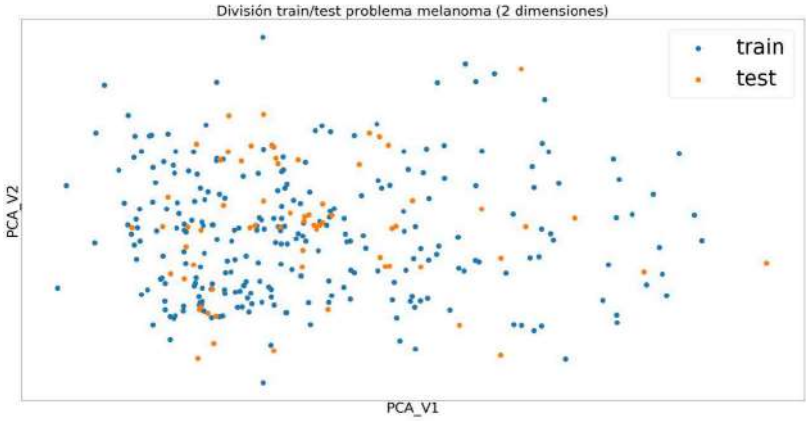
Tal como se indicó anteriormente, el procedimiento de entrenamiento en aprendizaje supervisado busca crear un modelo que se ajuste perfectamente a los datos que se dan como entrada. Para controlar si se realiza bien dicho ajuste, el algoritmo de construcción del modelo utiliza lo que se conoce como "*métrica de rendimiento*". Ésta determinará el error que se comete al realizar una predicción sobre las instancias del conjunto de entrenamiento con el modelo que se está generando.

Una vez concluido el aprendizaje, resulta de vital importancia realizar un proceso de validación del modelo. Esto significa determinar la calidad que mostrará el modelo cuando se ponga en funcionamiento sobre nuevos datos reales. Por este motivo, se utiliza el

llamado "conjunto de test", que deberá contener instancias nuevas que no se hayan utilizado durante la fase de entrenamiento.

De acuerdo a lo anterior, en aprendizaje supervisado existen de dos fases muy bien diferenciadas que son el **entrenamiento** (*train*) y la **validación** (*test*). Es necesario insistir que se deben utilizar dos conjuntos de datos totalmente independientes puesto que, de utilizar algún ejemplo de test durante el entrenamiento, se estará sobre-estimando la calidad del modelo generado.

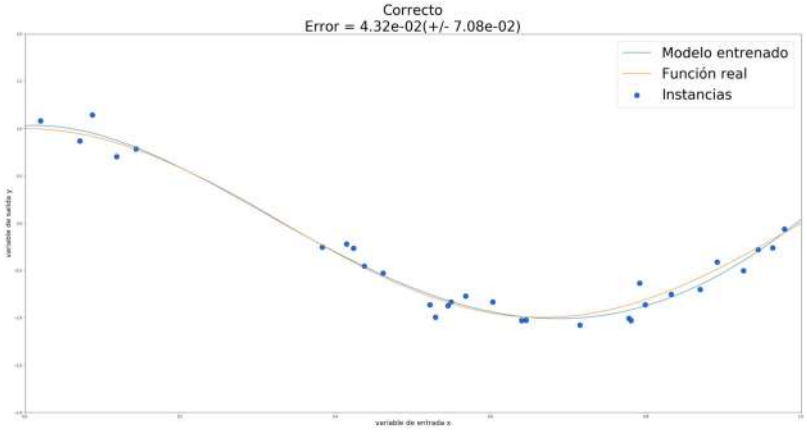
En la siguiente imagen se puede observar cómo se divide el conjunto de datos de entrada en dos grupos distintos, donde cada ejemplo está marcado con un color diferente si está dentro de "entrenamiento" (azul) o de "test" (naranja).



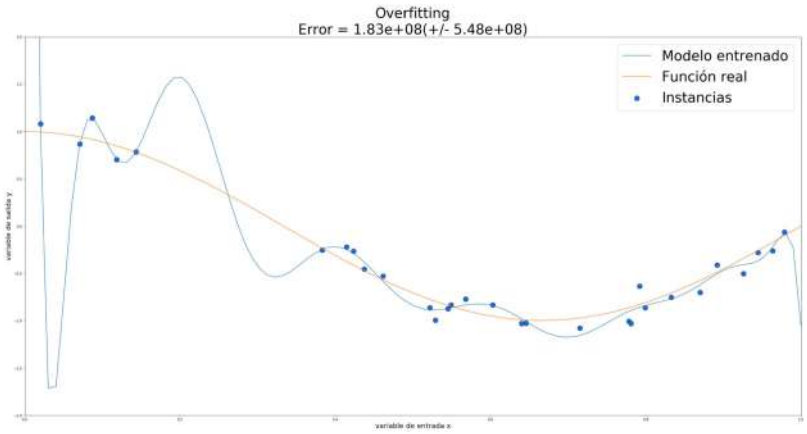
5.1 Casos de estudio en validación

A partir de los resultados de predicción obtenidos en entrenamiento y test, se calcula una métrica de rendimiento en cada conjunto. En este momento, se pueden dar varios escenarios posibles:

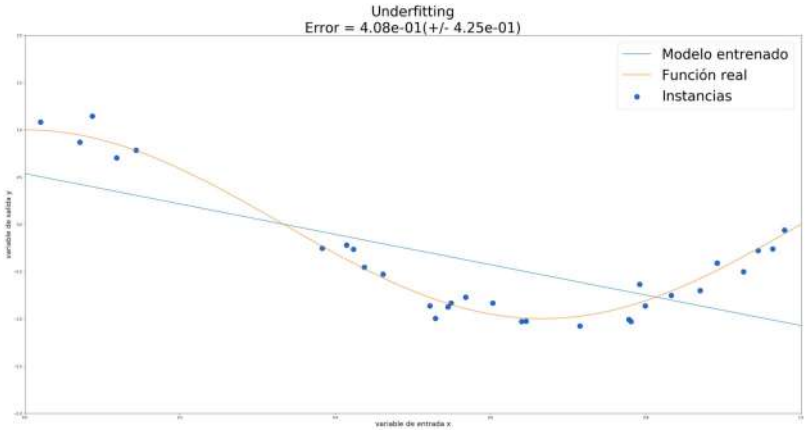
- Se obtiene una *alta calidad* de las métricas de rendimiento tanto en entrenamiento como test: el modelo es totalmente efectivo y se ha concluido el proceso.



- Se obtiene una *alta calidad* para entrenamiento, pero *muy baja* en test: se ha caido en el sobre-aprendizaje (*over-fitting*), es decir, se ha construido un modelo tan específico para los datos de entrenamiento, que luego es incapaz de generalizar bien con datos diferentes (test).



- Se obtiene una *baja calidad* tanto en entrenamiento como en test (*under-fitting*): el modelo no es bueno y se debe actuar mediante preprocesamiento, un ajuste adecuado de parámetros, o sencillamente alimentando al sistema con un mayor número de datos.



5.2 Tipos de técnicas de validación

Para dividir nuestro conjunto de datos original en los conjuntos de entrenamiento y test, existen tres metodologías diferentes:

- La más sencilla es la de retención o "**hold-out**", donde los datos originales se dividen en dos conjuntos disjuntos complementarios. Por ejemplo, un 60% de instancias iniciales para entrenamiento y el 40% restante para test.
- La más utilizada por su validez estadística es la validación cruzada de *k* particiones o "**k-fold cross validation**", donde se divide el conjunto original en *k* partes disjuntas. Por ejemplo, se separan las instancias en 5 grupos sin reemplazamiento.
- Existe la posibilidad de hacer una experimentación exhaustiva con "dejar uno fuera" o "**leave one out validation (LOOV)**". En este caso se utiliza todo el conjunto para entrenar, salvo un ejemplo para test.

En lo sucesivo, se explicarán con más detalle las principales de estas técnicas, poniendo especial atención a la más recomendada de todas: *la validación cruzada de k particiones*. La razón fundamental es que se emplean todas las instancias del conjunto de datos en las diferentes particiones de test, promoviendo un mayor rigor estadístico, y por tanto validez de los resultados obtenidos.

5.3 Validación Hold-out

Tal como se introdujo anteriormente, "**hold-out**" crea dos conjuntos simples, es decir, un fichero de entrenamiento y otro de test, mediante una partición en un porcentaje determinado de los datos. El porcentaje seleccionado para cada subconjunto queda a disposición del usuario, si bien valores típicos suelen estar por encima del 60, 75 u 80% para entrenamiento, y el resto para test.

La ventaja del método "**hold-out**" es que es muy sencillo y eficiente de realizar. Sin embargo, este método no es recomendable puesto que la calidad final del modelo dependerá en gran medida de cómo se hayan dividido los datos. En otras palabras, al separar las instancias de manera totalmente aleatoria podría darse el caso que todas las instancias seleccionadas para test sean las más difíciles de identificar.

Se utilizará por tanto **hold-out** cuando se realicen pruebas iniciales para conocer el comportamiento base de los diferentes modelos (algoritmos de Machine Learning) aplicados al problema bajo estudio.

A continuación, se va a conocer el código para crear las particiones de datos, donde todas las instrucciones están debidamente comentadas para entender su función concreta. El parámetro principal utilizado en este caso será el `ratio` para definir la división entre el conjunto de *test* y *entrenamiento* respectivamente.

Se recuerdan inicialmente algunos detalles sobre el código fuente mostrado:

- Como apoyo en todo el curso se utilizará la biblioteca Scikit-learn por lo que se importará toda la funcionalidad y módulos del mismo en las primeras líneas del código
- El particionamiento de datos se realiza por defecto de manera aleatoria. Por este motivo, se debe utilizar lo que se conoce como "semilla". Éste es un valor fijo que permite generar siempre los mismos resultados cada vez que se ejecute el código con dicha semilla indicada en el parámetro `random_state`.
- Siempre se utilizará como última instrucción un `print()` para mostrar un resultado que permita interpretar todo el proceso.

In [2]:
```
#se carga el codigo necesario para hacer "hold-out"
from sklearn.model_selection import train_test_split

#Parámetros usados:
rd = 42 #al ser un proceso no determinista, se fija una semilla
ratio = 0.2  #El parámetro más importante: cuál es el ratio usado para entrenamient

#A continuación se hace la división. Importante: se divide la entrada y salida (X e
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=ratio,random_st

#Se imprimen los índices de train y test con respecto al conjunto original
print("%s %s" % (X_train.index, X_test.index))
```

```
Int64Index([ 68, 181,  63, 248,  60,  15, 290, 137, 155, 517,
            ...
            330, 214, 466, 121,  20,  71, 106, 270, 435, 102],
           dtype='int64', length=455) Int64Index([204,  70, 131, 431, 540, 567, 36
9,  29,  81, 477,
            ...
            549, 530, 163, 503, 148, 486,  75, 249, 238, 265],
           dtype='int64', length=114)
```

Los siguientes fragmentos de código muestran el tamaño de los conjuntos que se acaban de crear

In [3]:
```
#Primero train (X e y)
X_train.shape, y_train.shape
```

Out[3]:
```
((455, 30), (455, 1))
```

In [4]:
```
#Luego test (x e y)
X_test.shape, y_test.shape
```

Out[4]:
```
((114, 30), (114, 1))
```

En los resultaod anteriores se puede observer que, al realizar una división en 80%-20% (se aplicó un `ratio = 0.2` para test) se tienen 455 instancias en entrenamiento ($569 \cdot 0.8$) y 114 para test ($569 \cdot 0.2$).

Es extremadamente importante que ambos conjuntos sean disjuntos, es decir, que ningún ejemplo de test se haya visto durante el entrenamiento, para poder realizar correctamente la validación. Este hecho se comprueba en el siguiente bloque de código, donde se "pregunta" si algún dato (instancia) del conjunto de entrenamiento (`X_train`) está contenido en el de test (`X_test`). El resultado deberá ser siempre `False` para todas las filas.

In [5]:
```
X_train.isin(X_test) #Instrucción que comprueba si algún elemento de test se repite

#La primera columna que se observa es el índice original, que está desordenado
```

Out[5]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry
68	False	False	False	False	False	False	False	False	False
181	False	False	False	False	False	False	False	False	False
63	False	False	False	False	False	False	False	False	False
248	False	False	False	False	False	False	False	False	False
60	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...
71	False	False	False	False	False	False	False	False	False
106	False	False	False	False	False	False	False	False	False
270	False	False	False	False	False	False	False	False	False
435	False	False	False	False	False	False	False	False	False
102	False	False	False	False	False	False	False	False	False

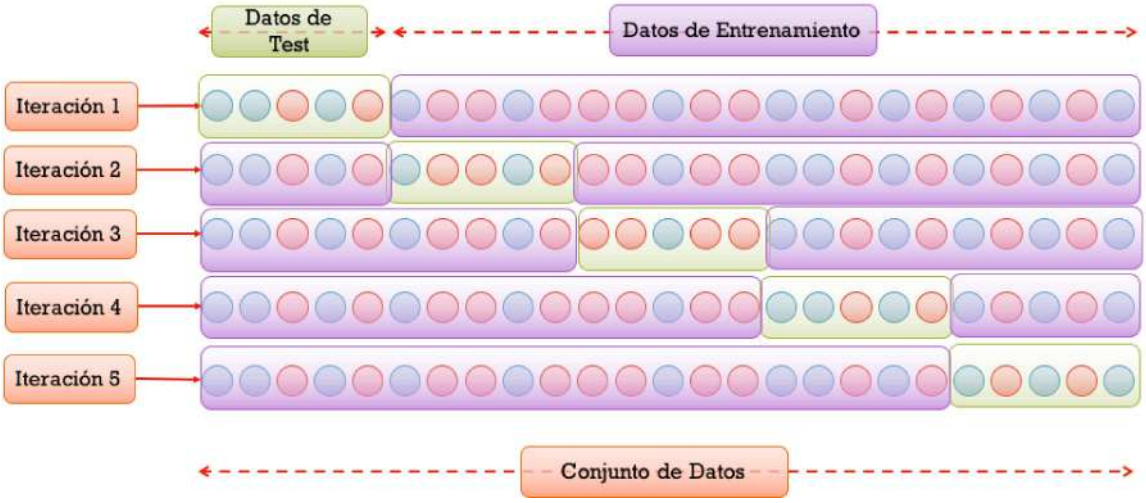
455 rows × 30 columns

5.4 Validación cruzada de k particiones

Esta aproximación es la más utilizada y recomendada cuando se realiza aprendizaje supervisado. El motivo principal es su rigor estadístico, así como el uso de la totalidad del conjunto de datos para entrenamiento y test, utilizando para ello un procedimiento iterativo, es decir, que se repitirá un número determinado de veces.

En concreto, cuando se utiliza una validación cruzada de k particiones se dividen los datos en k subconjuntos disjuntos, llamadas particiones. El objetivo, como se ha indicado anteriormente, es el de proceder a validar los modelos con diferentes combinaciones de dichas particiones. En concreto, las instancias pertenecientes a cada una de las *k* particiones se almacenerán en un conjunto diferente de test; mientras que la unión de las instancias que se encuentren en las *k-1* particiones restantes, se utilizarán para construir cada uno de los conjuntos de entrenamiento.

Como ejemplo, si se realiza una validación cruzada con 5 particiones (k=5), los datos se repartirían como se muestra en el esquema de la siguiente figura. Como se puede observar, para cada partición tomamos 1/5 de prueba, y los 4/5 restantes de entrenamiento.



A continuación, se indica cómo se puede realizar esta división de datos mediante Python, de nuevo aprovechando la funcionalidad de la biblioteca **Scikit-Learn**.

La principal diferencia del siguiente código con respecto al utilizado para *hold-out* reside en lo siguiente: en este caso, en lugar de generar directamente dos nuevas variables `train` y `test`, se calculan a priori los índices (posiciones) de las instancias que pertenecen a cada partición de test. Posteriormente, se utilizará una estructura repetitiva (bucle `for`) para generar las particiones realizando una búsqueda por índice (utilizando una función denominada `iloc`) sobre el conjunto original de entrada (`X`) y de salida (`y`).

La salida del ejemplo de código fuente mostrará, en cada línea, dos listas de índices (entre corchetes [ ]) representando las instancias seleccionadas para entrenamiento y test, respectivamente.

In [6]:
```
from sklearn.model_selection import KFold #cargamos las funciones necesarias

#Parámetros
rd = 42
particiones = 5

#Primera diferencia con hold-out, se crean a priori los índices de las particiones
```

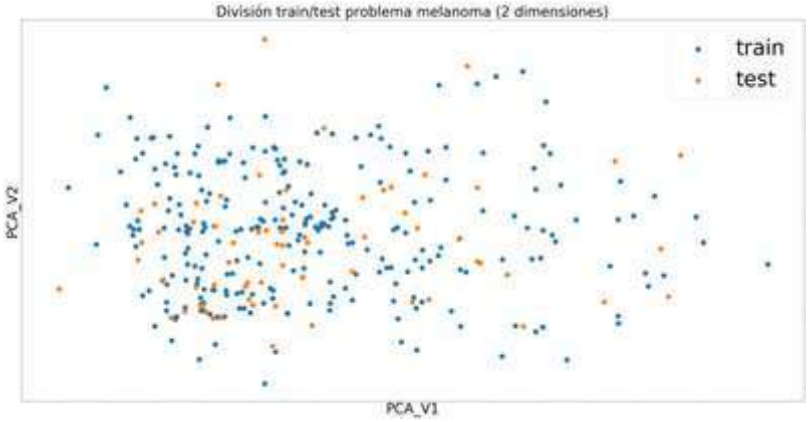
```
kf = KFold(n_splits=particiones,shuffle=True,random_state=rd)

#A continuación se realiza cada subdivisión.
#Notar que es necesario hacerlo iterativo (bucle for) para cada partición
for train, test in kf.split(X,y):
  print("%s %s" % (train, test))
  #Se crean los conjuntos iterativamente (iloc es una función de "búsqueda")
  X_train, X_test, y_train, y_test = X.iloc[train], X.iloc[test], y.iloc[train], y.
```

```
[  0   1   3   4   5   7   8  12  13  14  15  16  17  18  19  20  21  22
  23  24  25  26  27  28  31  32  33  34  35  36  37  38  40  41  42  43
  44  45  46  47  48  49  50  51  52  53  54  56  57  58  59  60  61  62
  63  64  65  66  67  68  69  71  74  80  85  87  88  89  91  92  93  94
  95  96  97  98  99 100 102 103 105 106 107 108 111 112 113 114 115 116
 117 119 120 121 122 123 124 125 126 127 128 129 130 133 134 135 136 137
 138 139 141 142 143 146 147 149 150 151 152 154 155 156 157 159 160 161
 162 164 166 168 169 170 171 172 173 174 175 176 178 179 180 181 183 184
 185 186 187 189 190 191 192 193 194 195 197 198 199 200 201 202 205 206
 207 209 210 212 213 214 215 216 217 218 219 220 221 223 224 225 226 227
 229 230 231 232 233 234 236 237 239 240 241 242 243 244 245 246 247 248
 251 252 253 254 256 258 259 260 261 262 263 266 267 268 269 270 272 273
 276 277 278 279 280 282 283 285 286 287 288 289 290 291 292 293 294 295
 296 297 298 299 300 301 302 303 304 306 307 308 309 310 311 312 313 314
 315 316 317 318 319 321 323 324 325 326 327 328 330 335 336 337 338 339
 340 341 342 343 344 345 346 347 348 349 350 351 352 354 355 356 357 358
 359 360 361 363 364 365 366 367 368 370 371 372 373 374 375 376 377 378
 379 381 383 385 386 387 388 389 390 391 392 396 397 398 399 400 401 402
 403 404 405 406 407 409 410 411 412 413 414 415 416 417 418 419 420 421
 423 426 427 428 429 430 432 433 434 435 436 437 438 439 440 442 443 444
 445 446 447 448 449 450 451 452 453 454 455 456 458 459 460 461 463 465
 466 467 469 470 471 472 473 474 475 476 478 479 480 481 483 484 485 487
 488 489 490 491 492 493 494 495 496 497 498 499 501 502 504 505 506 507
 508 509 510 512 513 514 515 516 517 518 519 521 522 523 524 525 529 533
 534 536 537 539 541 542 543 544 545 546 547 548 550 552 553 554 558 559
 560 562 563 566 568] [  2   6   9  10  11  29  30  39  55  70  72  73  75  76  77
78  79  81
  82  83  84  86  90 101 104 109 110 118 131 132 140 144 145 148 153 158
 163 165 167 177 182 188 196 203 204 208 211 222 228 235 238 249 250 255
 257 264 265 271 274 275 281 284 305 320 322 329 331 332 333 334 353 362
 369 380 382 384 393 394 395 408 422 424 425 431 441 457 462 464 468 477
 482 486 500 503 511 520 526 527 528 530 531 532 535 538 540 549 551 555
 556 557 561 564 565 567]
[  1   2   3   4   5   6   8   9  10  11  12  13  14  16  20  21  23  26
  27  28  29  30  32  34  35  36  37  38  39  40  41  43  44  45  47  48
  50  51  52  53  55  58  59  61  62  64  65  67  70  71  72  73  74  75
  76  77  78  79  80  81  82  83  84  85  86  87  90  91  92  94  95  96
  97  98  99 100 101 102 103 104 105 106 107 109 110 111 112 115 116 118
 119 120 121 122 123 125 127 128 129 130 131 132 133 134 135 136 138 139
 140 142 143 144 145 146 147 148 150 151 152 153 156 157 158 159 160 161
 162 163 164 165 166 167 168 169 170 171 175 177 178 179 180 182 183 186
 187 188 189 190 191 193 194 196 197 198 200 201 202 203 204 205 206 207
 208 211 212 213 214 215 216 217 219 220 221 222 223 224 225 226 228 229
 230 232 233 235 236 237 238 239 240 241 242 243 246 249 250 251 252 253
 254 255 256 257 258 259 260 262 263 264 265 266 267 269 270 271 273 274
 275 276 278 279 281 282 283 284 285 286 288 291 292 293 294 295 296 297
 299 300 302 303 305 306 307 308 309 312 313 314 315 316 317 318 320 321
 322 323 324 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340
 342 343 344 345 347 348 349 350 351 352 353 354 356 357 358 359 360 361
 362 363 365 366 367 368 369 370 371 372 373 375 376 377 378 379 380 382
 383 384 385 386 387 388 389 391 392 393 394 395 397 400 401 403 405 406
 408 409 412 413 415 416 417 418 419 420 422 423 424 425 427 429 430 431
 432 433 435 436 437 438 439 440 441 443 444 445 447 448 450 451 452 454
 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 471 472 473
 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491
 492 493 496 499 500 502 503 504 505 506 508 509 510 511 513 514 515 516
 518 519 520 521 522 524 525 526 527 528 529 530 531 532 533 534 535 536
 537 538 540 543 544 546 548 549 551 552 554 555 556 557 558 559 560 561
 563 564 565 566 567] [  0   7  15  17  18  19  22  24  25  31  33  42  46  49  54
56  57  60
  63  66  68  69  88  89  93 108 113 114 117 124 126 137 141 149 154 155
 172 173 174 176 181 184 185 192 195 199 209 210 218 227 231 234 244 245
 247 248 261 268 272 277 280 287 289 290 298 301 304 310 311 319 325 341
 346 355 364 374 381 390 396 398 399 402 404 407 410 411 414 421 426 428
```

```
 434 442 446 449 453 470 494 495 497 498 501 507 512 517 523 539 541 542
 545 547 550 553 562 568]
[  0   1   2   4   6   7   8   9  10  11  12  13  14  15  17  18  19  20
  21  22  24  25  27  28  29  30  31  32  33  34  35  39  40  41  42  43
  44  46  47  49  51  52  53  54  55  56  57  58  60  61  62  63  64  65
  66  68  69  70  71  72  73  75  76  77  78  79  80  81  82  83  84  85
  86  87  88  89  90  91  93  95  98  99 100 101 102 104 105 106 107 108
 109 110 112 113 114 115 117 118 120 121 124 125 126 127 128 129 130 131
 132 133 134 135 136 137 138 140 141 142 144 145 148 149 151 153 154 155
 156 158 159 160 161 162 163 164 165 166 167 169 170 171 172 173 174 176
 177 178 179 181 182 184 185 186 187 188 189 190 191 192 195 196 197 199
 200 201 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218
 219 221 222 223 224 226 227 228 230 231 232 233 234 235 236 238 240 241
 242 243 244 245 247 248 249 250 251 252 254 255 256 257 258 259 260 261
 264 265 267 268 269 270 271 272 273 274 275 276 277 279 280 281 282 283
 284 285 287 288 289 290 292 294 295 298 300 301 303 304 305 306 308 309
 310 311 313 314 315 317 319 320 322 325 326 327 329 330 331 332 333 334
 337 339 341 342 343 344 345 346 347 349 350 351 353 354 355 356 358 359
 361 362 364 366 369 371 372 373 374 376 378 379 380 381 382 383 384 385
 387 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405
 406 407 408 409 410 411 412 413 414 416 417 418 419 421 422 424 425 426
 427 428 429 430 431 434 435 437 438 441 442 443 445 446 447 448 449 451
 452 453 454 455 456 457 458 459 460 461 462 464 465 466 467 468 470 472
 474 475 476 477 478 479 482 483 484 486 488 491 492 493 494 495 497 498
 500 501 503 504 505 506 507 508 509 510 511 512 514 515 516 517 518 520
 522 523 524 525 526 527 528 529 530 531 532 534 535 537 538 539 540 541
 542 545 547 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563
 564 565 566 567 568] [  3   5  16  23  26  36  37  38  45  48  50  59  67  74  92
94  96  97
 103 111 116 119 122 123 139 143 146 147 150 152 157 168 175 180 183 193
 194 198 202 220 225 229 237 239 246 253 262 263 266 278 286 291 293 296
 297 299 302 307 312 316 318 321 323 324 328 335 336 338 340 348 352 357
 360 363 365 367 368 370 375 377 386 388 415 420 423 432 433 436 439 440
 444 450 463 469 471 473 480 481 485 487 489 490 496 499 502 513 519 521
 533 536 543 544 546 548]
[  0   1   2   3   5   6   7   8   9  10  11  13  14  15  16  17  18  19
  20  21  22  23  24  25  26  27  29  30  31  33  34  36  37  38  39  40
  42  43  45  46  48  49  50  52  54  55  56  57  58  59  60  62  63  64
  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83
  84  86  87  88  89  90  91  92  93  94  96  97  99 101 102 103 104 105
 106 108 109 110 111 113 114 116 117 118 119 121 122 123 124 126 128 130
 131 132 135 137 138 139 140 141 143 144 145 146 147 148 149 150 152 153
 154 155 156 157 158 160 161 162 163 165 166 167 168 172 173 174 175 176
 177 180 181 182 183 184 185 187 188 189 190 191 192 193 194 195 196 198
 199 201 202 203 204 205 207 208 209 210 211 212 214 216 217 218 220 222
 225 227 228 229 230 231 234 235 236 237 238 239 241 243 244 245 246 247
 248 249 250 251 252 253 255 257 259 260 261 262 263 264 265 266 268 269
 270 271 272 273 274 275 276 277 278 279 280 281 284 286 287 288 289 290
 291 293 295 296 297 298 299 300 301 302 303 304 305 307 308 309 310 311
 312 313 315 316 318 319 320 321 322 323 324 325 326 328 329 330 331 332
 333 334 335 336 337 338 339 340 341 343 344 345 346 348 350 352 353 355
 357 360 362 363 364 365 366 367 368 369 370 372 374 375 377 378 379 380
 381 382 384 385 386 387 388 389 390 391 393 394 395 396 398 399 401 402
 404 407 408 409 410 411 413 414 415 418 420 421 422 423 424 425 426 427
 428 431 432 433 434 435 436 439 440 441 442 444 445 446 449 450 453 454
 455 457 458 459 461 462 463 464 466 468 469 470 471 473 474 475 476 477
 478 480 481 482 483 484 485 486 487 489 490 491 493 494 495 496 497 498
 499 500 501 502 503 504 507 508 510 511 512 513 516 517 519 520 521 522
 523 526 527 528 529 530 531 532 533 535 536 537 538 539 540 541 542 543
 544 545 546 547 548 549 550 551 552 553 554 555 556 557 560 561 562 563
 564 565 566 567 568] [  4  12  28  32  35  41  44  47  51  53  61  65  85  95  98
100 107 112
 115 120 125 127 129 133 134 136 142 151 159 164 169 170 171 178 179 186
 197 200 206 213 215 219 221 223 224 226 232 233 240 242 254 256 258 267
```

```
 282 283 285 292 294 306 314 317 327 342 347 349 351 354 356 358 359 361
 371 373 376 383 392 397 400 403 405 406 412 416 417 419 429 430 437 438
 443 447 448 451 452 456 460 465 467 472 479 488 492 505 506 509 514 515
 518 524 525 534 558 559]
[  0   2   3   4   5   6   7   9  10  11  12  15  16  17  18  19  22  23
  24  25  26  28  29  30  31  32  33  35  36  37  38  39  41  42  44  45
  46  47  48  49  50  51  53  54  55  56  57  59  60  61  63  65  66  67
  68  69  70  72  73  74  75  76  77  78  79  81  82  83  84  85  86  88
  89  90  92  93  94  95  96  97  98 100 101 103 104 107 108 109 110 111
 112 113 114 115 116 117 118 119 120 122 123 124 125 126 127 129 131 132
 133 134 136 137 139 140 141 142 143 144 145 146 147 148 149 150 151 152
 153 154 155 157 158 159 163 164 165 167 168 169 170 171 172 173 174 175
 176 177 178 179 180 181 182 183 184 185 186 188 192 193 194 195 196 197
 198 199 200 202 203 204 206 208 209 210 211 213 215 218 219 220 221 222
 223 224 225 226 227 228 229 231 232 233 234 235 237 238 239 240 242 244
 245 246 247 248 249 250 253 254 255 256 257 258 261 262 263 264 265 266
 267 268 271 272 274 275 277 278 280 281 282 283 284 285 286 287 289 290
 291 292 293 294 296 297 298 299 301 302 304 305 306 307 310 311 312 314
 316 317 318 319 320 321 322 323 324 325 327 328 329 331 332 333 334 335
 336 338 340 341 342 346 347 348 349 351 352 353 354 355 356 357 358 359
 360 361 362 363 364 365 367 368 369 370 371 373 374 375 376 377 380 381
 382 383 384 386 388 390 392 393 394 395 396 397 398 399 400 402 403 404
 405 406 407 408 410 411 412 414 415 416 417 419 420 421 422 423 424 425
 426 428 429 430 431 432 433 434 436 437 438 439 440 441 442 443 444 446
 447 448 449 450 451 452 453 456 457 460 462 463 464 465 467 468 469 470
 471 472 473 477 479 480 481 482 485 486 487 488 489 490 492 494 495 496
 497 498 499 500 501 502 503 505 506 507 509 511 512 513 514 515 517 518
 519 520 521 523 524 525 526 527 528 530 531 532 533 534 535 536 538 539
 540 541 542 543 544 545 546 547 548 549 550 551 553 555 556 557 558 559
 561 562 564 565 567 568] [  1   8  13  14  20  21  27  34  40  43  52  58  62  64
71  80  87  91
  99 102 105 106 121 128 130 135 138 156 160 161 162 166 187 189 190 191
 201 205 207 212 214 216 217 230 236 241 243 251 252 259 260 269 270 273
 276 279 288 295 300 303 308 309 313 315 326 330 337 339 343 344 345 350
 366 372 378 379 385 387 389 391 401 409 413 418 427 435 445 454 455 458
 459 461 466 474 475 476 478 483 484 491 493 504 508 510 516 522 529 537
 552 554 560 563 566]
```

Tal como se indicó, la salida que debería mostrarse arriba son los índices de las diferentes instancias en entrenamiento y test. Es importante observar que para test no coinciden los números en ninguno de los 5 casos.



Además de lo anterior, es muy común repetir el proceso de particionamiento (división) en k-conjuntos un número determinado de veces para añadir mayor rigor estadístico a los resultados obtenidos. Esto significa que, como en el caso de *hold-out* el objetivo es eliminar

la dependencia del proceso de división aleatoria, que puede provocar que las instancias de test más difíciles siempre caigan en el mismo conjunto.

Este procedimiento descrito en el párrafo anterior se denomina como *repeated k-fold* y se muestra en el siguiente ejemplo de código:

In [ ]:
```
from sklearn.model_selection import RepeatedKFold

#Parámetros
rd = 42
particiones = 5
repeticiones = 3

#Muy similar al anterior, pero incluyendo un parámetro adicional de repeticiones
rkf = RepeatedKFold(n_splits=particiones, n_repeats=repeticiones, random_state=rd)
for index, (train, test) in enumerate(rkf.split(X,y)):
  if index % particiones == 0:
    print("Repetición ",(index // particiones)+1)
  print("%s %s" % (train, test))
  X_train, X_test, y_train, y_test = X.iloc[train], X.iloc[test], y.iloc[train], y.
```

Nota importante: El número total de particiones y / o iteraciones no se puede estimar a priori y dependerá principalmente del número de instancias de que dispongamos. El caso más común es utilizar 10 particiones por defecto y 3 repeticiones. Sin embargo, en caso de pocas instancias se puede bajar a 5 particiones y 3 ó 5 repeticiones.

5.5 Validación cruzada estratificada (solo clasificación)

Esta alternativa es una variación muy adecuada para clasificación del *k-fold* que devuelve particiones de tipo estratificadas. Este término significa que cada conjunto contiene aproximadamente el mismo porcentaje de muestras de cada clase objetivo que el conjunto completo.

Esto resulta imprescindible para evitar el posible sesgo en los modelos y obtener una conclusiones equívocas. Es especialmente relevante cuando se trabaja con datos no balanceados, es decir, con un porcentaje muy pequeño de muestras de una de las clases.

Tal como se indicó al comienzo de esta actividad, este caso de estudio se observa con frecuencia en problemas del contexto de biología y/o salud.

En el siguiente ejemplo, se implementa la solución utilizando esta nueva versión mejorada de *k-fold*. El formato de llamada es prácticamente equivalente al caso anterior de validación cruzada.

Para entender las diferencias en su comportamiento, se realiza una comparativa entre ambos esquemas de particionamiento, comprobando la distribución final de ejemplos que hay en cada clase. Para ello se utiliza una instrucción de `numpy` llamada `bincount` que simplemente cuenta el número de ocurrencias de cada valor dentro de la lista.

In [8]:
```
from sklearn.model_selection import StratifiedKFold, KFold
import numpy as np

#Parámetros
rd = 42
```

```
particiones = 5

#Realizamos partición estratificada
skf = StratifiedKFold(n_splits=particiones,shuffle=True,random_state=rd)

#Vamos a chequear las diferencias con KFold
kf = KFold(n_splits=particiones,shuffle=True,random_state=rd)

#A continuación hacemos cada subdivisión.
print("Comprobar distribución de clases en SKF:")
for train, test in skf.split(X,y):
  X_train, X_test, y_train, y_test = X.iloc[train], X.iloc[test], y.iloc[train], y.
  print('train -  {}   |   test -  {}'.format(np.bincount(y_train.iloc[:,0]), np.bi

print("\nComprobar distribución de clases en KF:")
for train, test in kf.split(X, y):
  X_train, X_test, y_train, y_test = X.iloc[train], X.iloc[test], y.iloc[train], y.
  print('train -  {}   |   test -  {}'.format(np.bincount(y_train.iloc[:,0]), np.bi
```

```
Comprobar distribución de clases en SKF:
train -  [169 286]   |   test -  [43 71]
train -  [169 286]   |   test -  [43 71]
train -  [170 285]   |   test -  [42 72]
train -  [170 285]   |   test -  [42 72]
train -  [170 286]   |   test -  [42 71]

Comprobar distribución de clases en KF:
train -  [169 286]   |   test -  [43 71]
train -  [175 280]   |   test -  [37 77]
train -  [169 286]   |   test -  [43 71]
train -  [169 286]   |   test -  [43 71]
train -  [166 290]   |   test -  [46 67]
```

En el ejemplo anterior se ha mostrado la distribución de instancias en las dos clases para las particiones de test. A pesar que el conjunto de datos de *breast cancer* no presenta un alto desequilibrio de clases, sí que se ha observado que se puede crear sin quererlo un sesgo hacia alguna de las mismas (por ejemplo en la segunda de las particiones).

5.6 Validación "Leave One Out" (LOOV)

Esta es, junto con la técnica *hold-out*, la más sencilla de las técnicas de validación. En este caso concreto, cada muestra se considera un único conjunto de prueba, mientras que las muestras restantes forman el conjunto de entrenamiento.

LOOV debe utilizarse con especial cuidado. En concreto, en lugar de construir *k* modelos (como en la validación cruzada k-fold), ahora se obtienen *n* modelos, siendo *n* el tamaño del conjunto de datos. Teniendo en cuenta lo anterior, y dado que cada modelo se aprende a partir del conjunto de datos completo (menos una muestra), el coste computacional de esta validación es simplemente enorme.

Los usuarios deben tener en cuenta que, en lo que respecta a las métricas de rendimiento, esta técnica de validación ofrece una alta varianza. Este comportamiento es esperable ya que todos los modelos son básicamente iguales (se construyeron con casi las mismas instancias).

LOOV se puede simular a partir de `KFold(n_splits=n)` con `n` el número total de muestras del conjunto de datos.

In [9]:
```
#Parámetros
rd = 42
particiones = len(X)

#Parámetros rd = 42 particiones = todas las muestras
loov = KFold(n_splits=particiones,shuffle=True,random_state=rd)

#A continuación hacemos cada subdivisión
for train, test in loov.split(X,y):
  X_train, X_test, y_train, y_test = X.iloc[train], X.iloc[test], y.iloc[train], y.
  #A continuación se debe realizar el aprendizaje y la validación

print("Ejemplo última partición: Total sizeTr[{}] sizeTst[{}]".format(len(y_train),
```

```
Ejemplo última partición: Total sizeTr[568] sizeTst[1]
```

REFERENCIAS BIBLIOGRÁFICAS

- Han, J., Kamber, M., Pei, J. (2011). Data Mining: Concepts and Techniques. San Francisco, CA, USA: Morgan Kaufmann Publishers. ISBN: 0123814790, 9780123814791
- Hastie, T., Tibshirani, R., Friedman, J. H. (2001). The Elements of Statistical Learning. Springer Verlag.
- Witten, I. H., Frank, E., Hall, M. A., Pal, C. J. (2017). Data mining: practical machine learning tools and techniques. Amsterdam; London: Morgan Kaufmann. ISBN: 9780128042915 0128042915
- Molnar, C. (2023). Interpretable Machine Learning A Guide for Making Black Box Models Explainable. https://christophm.github.io/interpretable-ml-book/ (consultado el 26 de octubre de 2023)
- Scikit-Learn: Cross-validation: evaluating estimator performance https://scikit-learn.org/stable/modules/cross_validation.html (visitado el 25 de Junio de 2020)

Referencias adicionales

- Alpaydin, E. (2016). Machine Learning: The New AI. MIT Press. ISBN: 9780262529518
- James, G., Witten, D., Hastie, T., Tibshirani, R. (2013). An introduction to statistical learning (Vol. 112). Springer.
- Kohavi, R. (1995). A Study of Cross-Validation and Bootstrap for Accuracy Estimation nd Model Selection. Proceedings of the International Joint Conference on Artificial Intelligence IJCAI), 1995 (p./pp. 1137--1145).
- Rao, R. B. & Fung, G. (2008). On the Dangers of Cross-Validation. An Experimental Evaluation. SDM (p./pp. 588-596), : SIAM. ISBN: 978-1-61197-278-8