TDVI: Inteligencia Artificial

Overfitting & Validación de Modelos

UTDT - LTD

Estructura de la clase

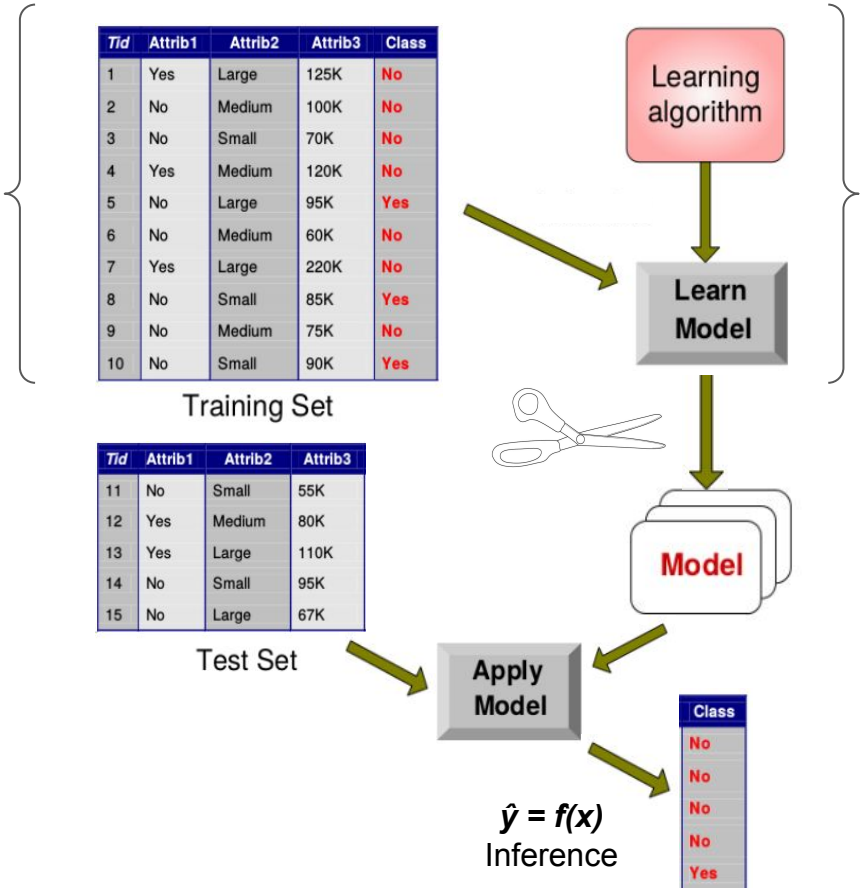
- Repaso de Aprendizaje Supervisado
- Overfitting y underfitting
- Validación de modelos

Estructura de la clase

- Repaso de Aprendizaje Supervisado
- Overfitting y underfitting
- Validación de modelos

Repaso de Aprendizaje Supervisado

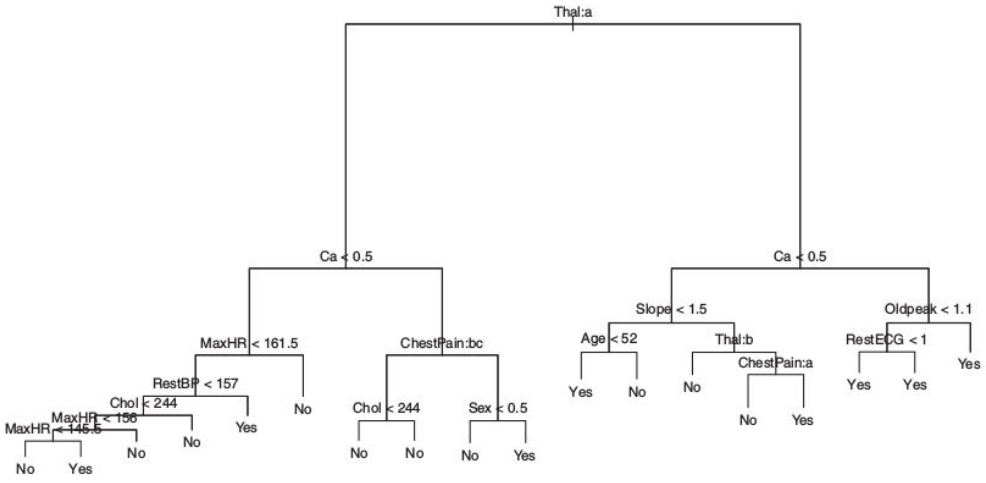
Esquema general de aprendizaje supervisado

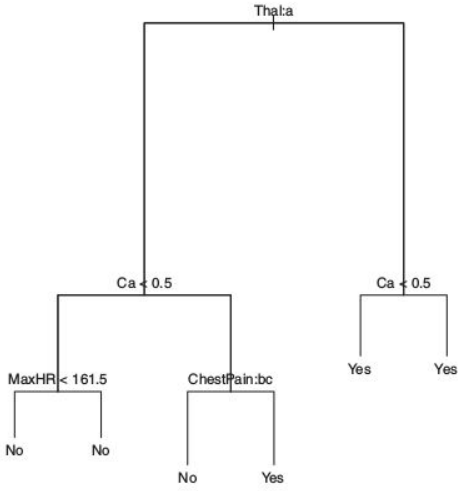


¡El modelo depende tanto de los datos de entrenamiento como del algoritmo de aprendizaje utilizado!

Repaso de Aprendizaje Supervisado

Distintos modelos pueden tener mayor o menor flexibilidad. Incluso para un mismo algoritmo de aprendizaje esto ocurre





Repaso de Aprendizaje Supervisado

Animal	¿Da a luz?	¿Vuela?	¿Vive en el agua?	¿Tiene piernas?	¿Mamífero?
humano	sí	no	no	sí	mamífero
pitón	no	no	no	no	no-mamífero
salmón	no	no	sí	no	no-mamífero
ballena	sí	no	sí	no	mamífero
rana	no	no	a veces	sí	no-mamífero
dragón de komodo	no	no	no	sí	no-mamífero
murciélago	sí	sí	no	sí	mamífero
paloma	no	sí	no	sí	no-mamífero
gato	sí	no	no	sí	mamífero
tiburón leopardo	sí	no	sí	no	no-mamífero
tortuga	no	no	a veces	sí	no-mamífero
pingüino	no	no	a veces	sí	no-mamífero
puercoespín	sí	no	no	sí	mamífero
anguila	no	no	sí	no	no-mamífero
salamandra	no	no	a veces	sí	no-mamífero
monstruo de gila	no	no	no	sí	no-mamífero
ornitorrinco	no	no	no	sí	mamífero
búho	no	sí	no	sí	no-mamífero
delfín	sí	no	sí	no	mamífero
águila	no	sí	no	sí	no-mamífero

¿Da a luz?	¿Vuela?	¿Vive en el agua?	¿Tiene piernas?	¿Mamífero?
sí	no	sí	no	¿?

En clases se entrenó un modelo que predice una probabilidad igual a 0.8 de que un animal con estas características sea mamífero

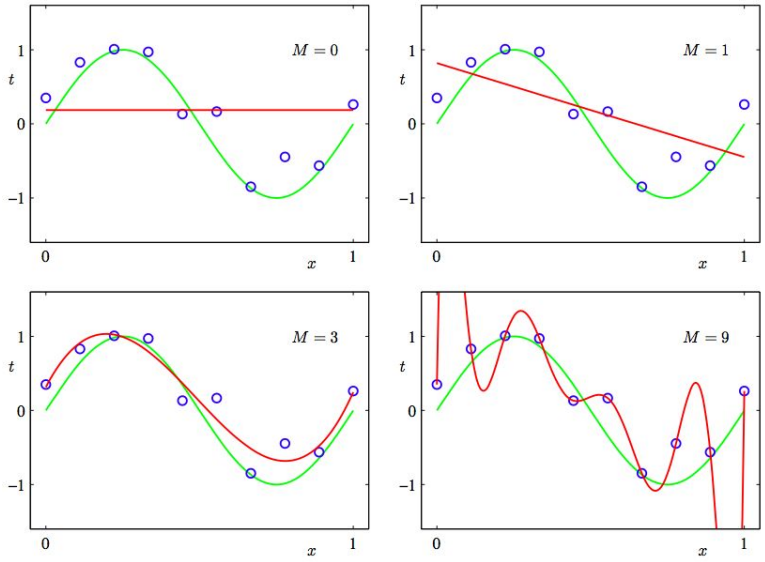
¿Es una buena predicción?

Estructura de la clase

- Repaso de Aprendizaje Supervisado
- Overfitting y underfitting
- Validación de modelos

Overfitting y underfitting

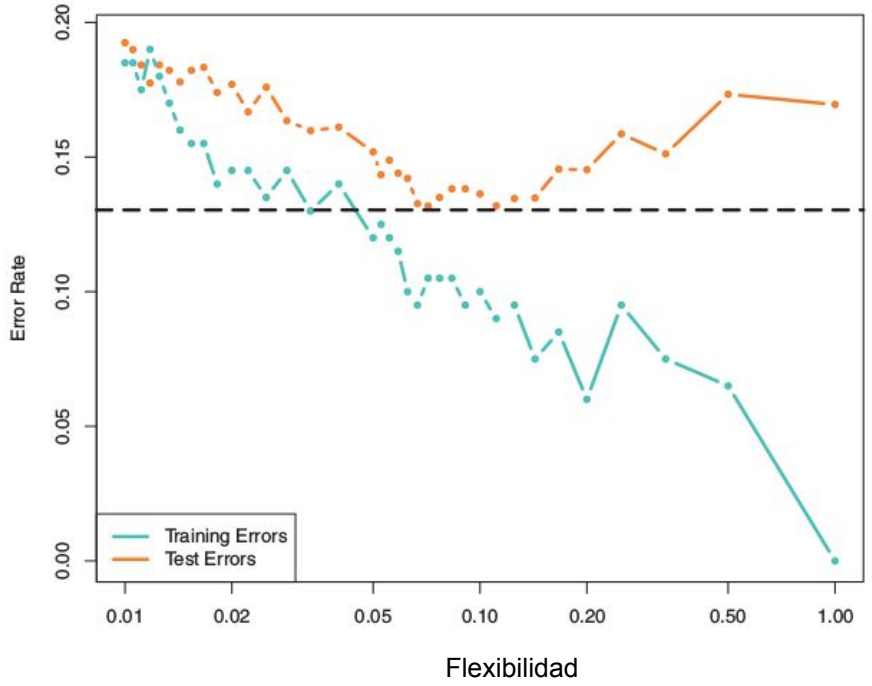
Veamos la notebook *underfitting_and_overfitting.ipynb* para ganar intuiciones



Underfitting: el modelo es demasiado rígido y no llega ni a capturar los patrones relevantes de entrenamiento, tendrá mala performance tanto en training como en testing

Overfitting: el modelo es demasiado flexible, ajusta en exceso las particularidades de los datos de entrenamiento, tendrá buena performance en training y mala performance en testing (la cual puede mejorarse quitándole flexibilidad)

Overfitting y underfitting



¿Underfitting o overfitting?

¿Underfitting o overfitting?

Overfitting y underfitting

Al momento de modelar, uno tiene muchas decisiones que tomar, por ej.:

- Qué datos tomar como inputs para entrenar modelo
- Qué preprocesamiento de los datos hacer
- Qué algoritmo de aprendizaje usar para predecir (*no free-lunch theorem*)
- Con qué valores de hiperparámetros entrenar el algoritmo de aprendizaje

Cada decisión seguramente impacte sobre la performance predictiva del modelo. Por eso, elegir un modelo que tenga buena performance en datos desconocidos es una tarea compleja

Estrategia: de alguna manera poder estimar la performance de un modelo cualquiera sobre nuevas observaciones. Si se predice una mejor performance para un modelo *i* que para uno *j*, se elige a *i* por sobre *j*

Estructura de la clase

- Repaso de Aprendizaje Supervisado
- Overfitting y underfitting
- Validación de modelos

Validación de modelos

Esquema que seguiremos a lo largo de la materia:

Setup:

- Existe un proceso generador de datos (***P(X, y)***). No lo conocemos
- Disponemos de instancias/observaciones generadas por dicho proceso. Las llamamos **train set** (X_{tr} e y_{tr}). OJO: podríamos haber tenido otro train set (primer componente aleatorio)
- Existe una relación del tipo $\boldsymbol{y = f(X) + \varepsilon}$. En donde 1) no conocemos *f* y en donde 2) ε es ruido irreducible (se cumple que $\boldsymbol{E(\varepsilon)=0}$)

Objetivo:

- Tendremos nuevas instancias (generadas por el mismo proceso) para las cuales sólo conoceremos *X*. Queremos estimar bien *y*, Las llamaremos **test set** (X_{ts} e y_{ts}). OJO: podríamos haber tenido otro test set (segundo componente aleatorio)

Estrategia

- Sobre la base de X_{tr} e y_{tr}, entrenamos un modelo de aprendizaje supervisado (e.g., un árbol) y obtenemos una estimación de f. Llamamos **f** al modelo predicho

Validación de modelos

¿Dado un modelo propuesto (e.g., un árbol de profundidad máxima 5) qué genera incertidumbre en la performance en testing?

Que el train set y el test set sean instancias de un proceso aleatorio genera, de antemano, incertidumbre respecto a los valores de las medidas de performance. Supongamos que pudiéramos "resamplear" el train set o el test set.

- Otras instancias de testing → un mismo f predice sobre instancias distintas → la performance puede variar (e.g., por ahí ahora son instancias simples/difíciles)

- Otras instancias de training → se obtiene otro f → varían las predicciones (incluso para idénticos X_{ts} e y_{ts}) → la performance puede variar (e.g., quizás, por puro azar, las nuevas instancias son poco representativas)

Validación de modelos

Esta conceptualización da a lugar a dos formas de pensar la performance/error del modelo:

- **Test error** (o generalization error): dado un training set *T*, es el error (*L*) esperado para una nueva observación de test independiente:

$$\mathrm{Err}_{\mathcal{T}} = \mathrm{E}[L(Y, \hat{f}(X))|\mathcal{T}]$$

- **Expected prediction error** (o expected test error): es el error esperado para una nueva observación de testing independiente teniendo en cuenta todo lo que es aleatorio (incluye la aleatoriedad intrínseca del proceso que genera X_{tr} e y_{tr})

$$\mathrm{Err} = \mathrm{E}[L(Y, \hat{f}(X))] = \mathrm{E}[\mathrm{Err}_{\mathcal{T}}]$$

Idealmente, queremos conocer el test error (Err_T). En la práctica tenderemos a estimar algo más cercano al expected prediction error (Err)

Validación de modelos

Probémoslo en Python. Veamos el primer bloque de código

Validación de modelos

Queremos predecir bien sobre nuevas instancias, pero al no disponer de las mismas, no sabemos cómo un modelo predecirá sobre ellas. Entonces estimaremos/predeciremos el error del modelo (IMPORTANTE: es otro problema de predicción, diferente)

¿Cómo podemos encarar el problema de guiar la búsqueda de un buen modelo en términos predictivos?

Estrategia: simularemos de alguna manera la división entre "datos conocidos" y "datos desconocidos"

Utilizaremos los "datos conocidos" para entrenar al modelo (que tendrá determinadas características, e.g., variables, valor de hiperparámetros, etc.)

Utilizaremos los "datos desconocidos" para estimar la performance del modelo sobre nuevos datos. De esta forma validamos nuestras decisiones (de aquí que a este conjunto de datos se le suela llamar validation set)

Validación de modelos - Validation/Holdout Set

La manera más simple de simular el comportamiento sobre datos desconocidos es separar una submuestra al azar de observaciones del training set como conjunto de validación

FIGURE 5.1. *A schematic display of the validation set approach. A set of n observations are randomly split into a training set (shown in blue, containing observations 7, 22, and 13, among others) and a validation set (shown in beige, and containing observation 91, among others). The statistical learning method is fit on the training set, and its performance is evaluated on the validation set.*

¿Una vez definido el modelo final, qué observaciones usamos para entrenarlo?

Idealmente todas (pero esto no siempre se hace. E.g., en redes neuronales profundas… ¿por qué?)

Validación de modelos - Validation/Holdout Set

Potenciales problemas (como estimador de performance futura):

- La estimación de performance depende de la participación de observaciones en training y validation (la predicción de performance tiene variabilidad). Una forma de disminuir la variabilidad es repetir el proceso muchas veces y promediar (*repeated validation set*)

- A mayor cantidad de datos, se espera mejor performance del modelo → Si el conjunto de entrenamiento quedase con pocas observaciones, se podría tener una estimación de performance pesimista (no parece tan grave si uno tiene muchísimos datos)

Validación de modelos - Validation/Holdout Set

Validación de modelos - Leave-one-out Cross-validation

Una alternativa que no implica tener que reducir tanto el tamaño del conjunto de entrenamiento

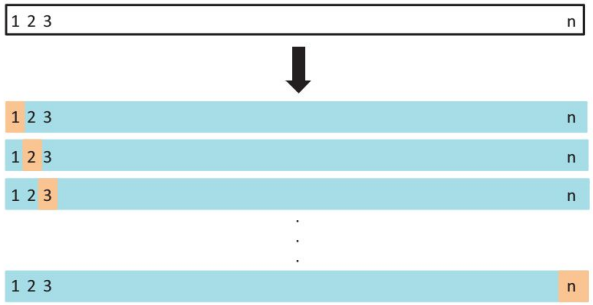


FIGURE 5.3. *A schematic display of LOOCV. A set of n data points is repeatedly split into a training set (shown in blue) containing all but one observation, and a validation set that contains only that observation (shown in beige). The test error is then estimated by averaging the n resulting MSE's. The first training set contains all but observation 1, the second training set contains all but observation 2, and so forth.*

$$CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n} Perf_i$$

Ventajas:

- Los resultados no dependen de una partición al azar
- Cada uno de los modelos es entrenado con prácticamente todas las observaciones
- Toda observación es usada una única vez para validar

Desventajas:

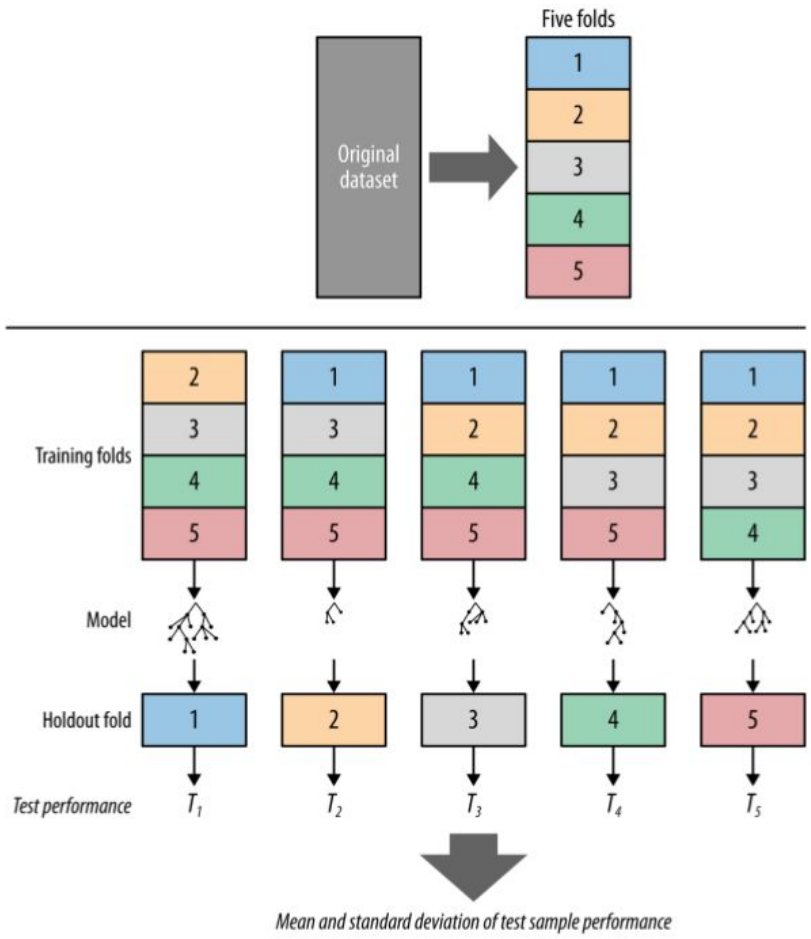
- Requiere de mucho computo (sólo sirve cuando uno tiene poco datos)
- El estimador tiene propiedades estadísticas no deseadas (ISLP 5.1.4)

Validación de modelos - Leave-one-out Cross-validation

Probémoslo en Python. Veamos el tercer bloque de código

Validación de modelos - k-fold cross-validation

Una alternativa "intermedia" es usar k-fold cross-validation



- Típicamente los valores de *k* que se utilizan son 3, 5 ó 10
- Como el resultado también depende de particiones al azar, para mejorar la estimación de la performance en testeo, a veces se repite el proceso varias veces

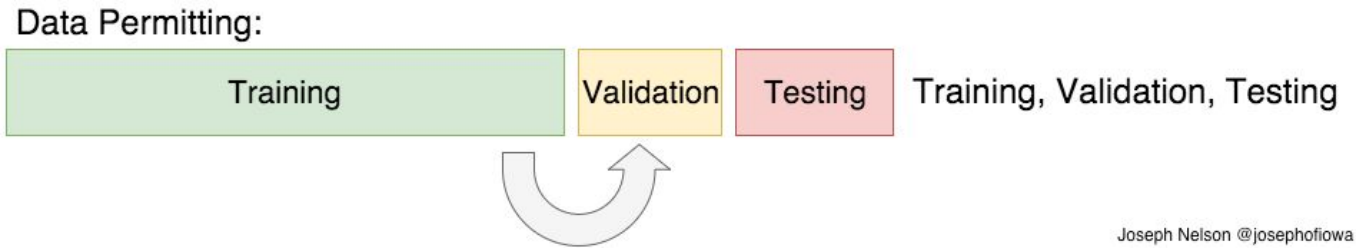
¿A qué se parece si *k* = *n*? ¿A qué se parece si *k* = *2*?

Validación de modelos - k-fold cross-validation

Probémoslo en Python. Veamos el cuarto bloque de código

Model selection - training, validation y testing sets

Si uno realiza múltiples validaciones, es posible "overfitear" el conjunto de validación/holdout. Por este motivo, muchas veces se trabaja con tres conjuntos: 1) entrenamiento, 2) validación y 3) testeo



Comúnmente se divide en las siguientes proporciones:

- Training 60% (80%)
- Validation 20% (10%)
- Testing 20% (10%)

Model selection: consiste en estimar la performance de diferentes modelos para elegir el mejor

Model assessment: después de haber elegido un modelo final, intentar estimar su performance en nuevos datos (*generalization error*)

Validación de modelos - k-fold cross-validation

Probémoslo en Python. Veamos el quinto bloque de código

Validación de modelos

Los métodos propuestos (para estimar performance) suponen que las observaciones son *i.i.d.*, lo cual puede no ser cierto

- Ejemplo de no idénticamente distribuidos

Se dispone de datos de clientes de tres países (A, B y C) para predecir churn. Los clientes de cada país tienen características propias, aunque dentro de un mismo país, estos son independientes entre sí. Se sumarán clientes de un país D y se quiere saber qué performance tendrá un modelo entrenado con los datos de A, B y C para predecir churn en D

- Mal enfoque: mezclar las observaciones de A, B y C y reservar un porcentaje de las mismas como conjunto de validación
- Leave-one-group-out cross validation (mejor enfoque): entrenar un modelo con A y B, y validarlo con C (replica mejor lo que efectivamente sucederá)

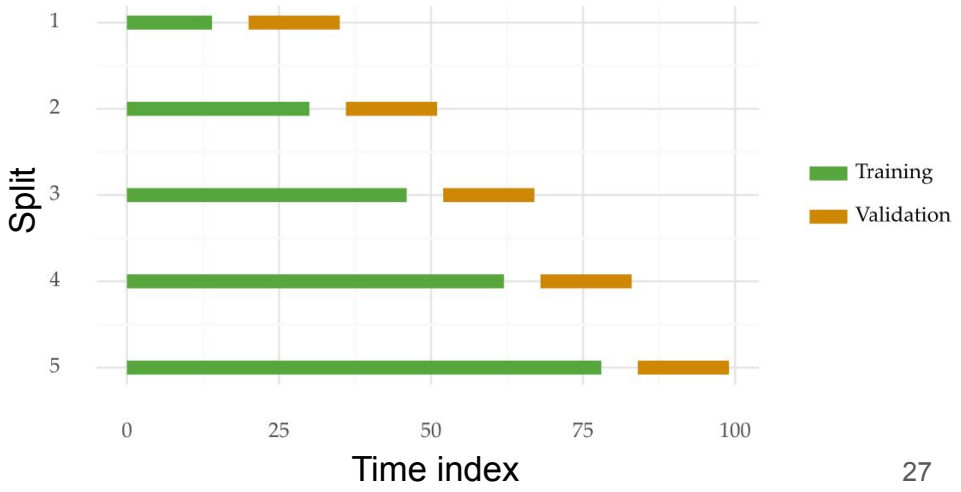
Validación de modelos

Los métodos propuestos (para estimar performance) suponen que las observaciones son *i.i.d.*, lo cual puede no ser cierto

- Ejemplo de no independientes

Se quiere generar un modelo que a partir de datos meteorológicos disponibles en un día dado, prediga la temperatura del día siguiente. Noten que incluso para una misma época del año, la temperatura de un día *t* impacta sobre la de *t+1* (e.g., el calor del pìso de un día afecta en el siguiente día)

- Mal enfoque: realizar un holdout set tradicional. Irreal: habrá observaciones de validación más viejas que las de entrenamiento
- Time-series cross validation (o alguna otra variante que considere temporalidad)



Bibliografía

- ISLP. Capítulo 5 (5.2. se verá más adelante en la materia)

Avanzada

- Hastie, Tibshirani & Friedman, "The Elements of Statistical Learning", Capítulo 7 (salvo secciones 7.6, 7.7, 7.8 y 7.9)