

The background image shows a modern university building with a light-colored stone facade and large glass windows. A prominent feature is a balcony with a metal railing on the left side. The text "IGMAT" is visible on the building's facade.

# Conceptos generales de aprendizaje supervisado

Curso de aprendizaje automático para  
**IGMAT** el INE

Alberto Torres Barrán

2020-02-02

# ¿Qué es el Aprendizaje Automático?

De la Wikipedia:

*Machine learning is a subfield of **computer science** that evolved from the study of **pattern recognition** and computational learning theory in artificial intelligence. In 1959, Arthur Samuel defined machinelearning as a “Field of study that gives computers the ability to learn without being **explicitly programmed**”. Machine learning explores the study and construction of algorithms that can learn from and make predictions on **data**.*

THIS IS YOUR MACHINE LEARNING SYSTEM?

YUP! YOU POUR THE DATA INTO THIS BIG  
PILE OF LINEAR ALGEBRA, THEN COLLECT  
THE ANSWERS ON THE OTHER SIDE.

WHAT IF THE ANSWERS ARE WRONG?

JUST STIR THE PILE UNTIL  
THEY START LOOKING RIGHT.



Fuente: [xkcd #1838](#)

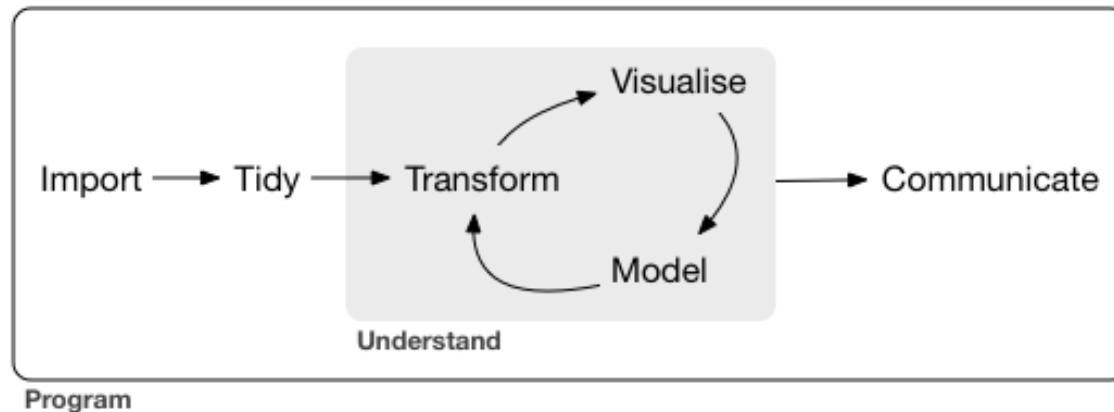
Data Science Data Mining

# Statistics

Artificial Intelligence

# Machine Learning

# Data science

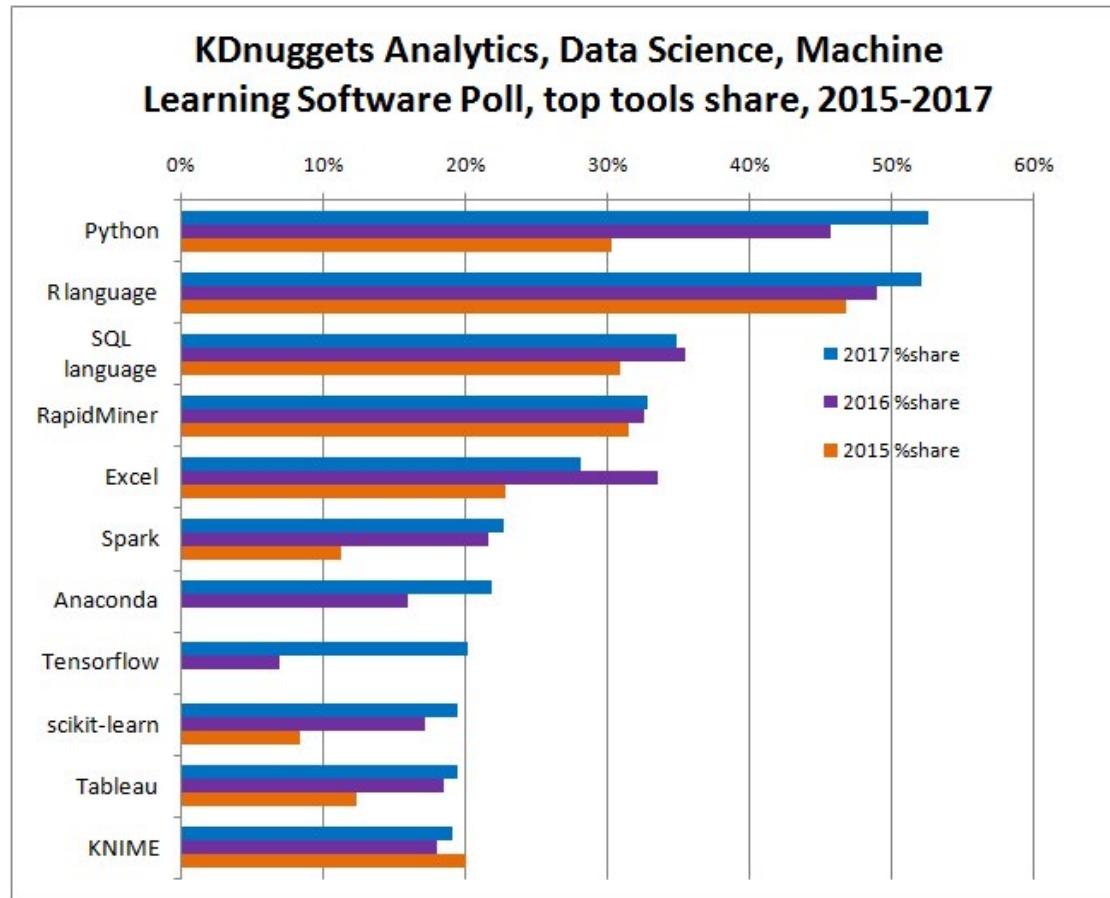


Fuente: [R for Data Science](#)

# Casos de éxito

- Coches autónomos
- Análisis de imágenes médicas
- Procesamiento de lenguaje natural
- AlphaGo y juegos Atari
- Generación de imágenes
- Sistemas de recomendación

# Herramientas



# Tipos de aprendizaje

Existen diversos tipos de tareas, dependiendo de la información disponible:

- **supervisado**: tenemos acceso a pares de ejemplos entrada-salida
- **no supervisado**: no tenemos acceso a las salidas
- otros (limitando de alguna forma el acceso a las salidas):
  - *activo*: el algoritmo puede acceder a la salida para nuevos datos de entrada
  - *semi-supervisado*: solo se tienen salidas para algunos datos
  - *refuerzo*: no se tiene el valor de la salida, pero si una indicación de lo lejos o cerca que se encuentra

# Referencias

1. Jerome H. Friedman. [Data Mining and Statistics: What's the Connection?](#) (1998)
2. Leo Breiman. [Statistical Modeling: The Two Cultures](#) (2001)
3. Cross Validated. [What is the difference between data mining, statistics, machine learning and AI](#) (2010).
4. Sakthi Dasan Sekar. [What is the difference between Artificial Intelligence, Machine Learning, Statistics, and Data Mining](#) (2014)
5. Cross Validated. [What exactly is Big Data?](#) (2015)
6. David Donoho. [50 years of Data Science](#) (2015)

# Aprendizaje supervisado

- Tenemos disponibles datos con múltiples observaciones:
  - ejemplos (*examples*)
  - muestras (*samples*)
  - ...
- Varias variables por observación:
  - predictores
  - atributos (*attributes*)
  - características (*features*)
  - covariables (*covariates*)
  - variables independientes
  - variables explicativas
  - ...
- Una de ellas es de especial interés:
  - variable respuesta
  - variable dependiente
  - objetivo (*target*)
  - salida (*output*)
  - etiqueta (*label*)
  - ...

# Objetivos

1. Predecir el valor de la variable respuesta para nuevas observaciones
2. Obtener información sobre la relación entre las variables independientes y la salida

# Tipos de problemas

1. Regresión, si la variable respuesta es continua
2. Clasificación, si la variable respuesta es discreta
3. Otros: por ejemplo,
  - salida continua pero valores enteros
  - salida discreta pero los valores tienen un orden

# Aprendizaje estadístico

## Datos:

- Espacio de las muestras de entrada:
- Conjunto de posibles salidas:
- Conjunto de **entrenamiento**:  $S = \{x_i, y_i\}_{i=1}^n$ , contenido en el espacio  $\times$

## Objetivo:

- Aprender una regla de predicción (hipótesis),  $h : \quad \rightarrow$

## Asumimos:

- Los ejemplos se han generado por una distribución de probabilidad desconocida
- Existe una función de pérdida  $L : \quad \times \quad \rightarrow$  que mide como de lejos se encuentra  $h(x)$  de  $y$
- El conjunto posible de hipótesis ( $\quad$ ) es finito

# Minimización del riesgo empírico

Elegir  $h$  tal que minimice el riesgo esperado

$$R(h) = \int_{\times} L(h(x), y) dP(x, y)$$

**Problema:** cómo podemos calcular  $R$  si  $P$  es desconocida?

Podemos evaluar la función de pérdida en el conjunto  $S$  (riesgo empírico):

$$\hat{R}(h) = \frac{1}{n} \sum_{i=1}^n L(h(x_i), y_i)$$

Si  $n$  suficientemente grande, esperamos que  $\hat{R}(x) \sim R(x) \rightarrow$  minimizar el riesgo empírico es una buena aproximación de minimizar el riesgo esperado

# Descomposición del error

Minimizador del riesgo:

$$h^* = \arg \min_{h \in \mathcal{H}} R(h)$$

Minimizador del riesgo empírico:

$$\hat{h}^* = \arg \min_{h \in \mathcal{H}} \hat{R}(h)$$

Riesgo de Bayes o error de Bayes:

$$R^* = \inf_h R(h)$$

*Nota:* sobre todas las funciones  $h : \mathcal{X} \rightarrow \mathcal{Y}$ , no solo las contenidas en  $\mathcal{H}$  !!

La diferencia entre el riesgo y el error de Bayes es:

$$R(h) - R^* = \underbrace{(R(h) - R(\hat{h}^*))}_{\text{error optimización}} + \underbrace{(R(\hat{h}^*) - R(h^*))}_{\text{error estimación}} + \underbrace{(R(h^*) - R^*)}_{\text{error aproximación}}$$

**Error optimización:** como de buena es la optimización que llevó a la hipótesis  $h$ , relativa a al óptimo del riesgo empírico

- Disminuye al mejorar el algoritmo de optimización

**Error de estimación:** surge por aproximar el riesgo esperado con el riesgo empírico

- Disminuye si aumentamos el conjunto de datos de entrenamiento  $n$

**Error de aproximación:** surge por aproximar la mejor función posible por la mejor función dentro de

- Disminuye si reemplazamos por otra clase más flexible

# Ejemplo

- Elegimos como las clase de funciones del tipo  $f(x) = w_0 + x^T w$
- Función de pérdida:  $L(y, f(x)) = (y - f(x))^2$
- Riesgo empírico:

$$R(w) = \frac{1}{n} \sum_{i=1}^n (y_i - w_0 + x_i^T w)^2$$

# Regresión lineal

Dado el conjunto de entrenamiento  $S = \{y_i, x_i\}_{i=1}^n$

- Agrupamos todos los ejemplos de entrada  $x_i$  en una matrix de tamaño  $n \times d$
- Agrupamos todas las salidas en un vector columna  $y$  de tamaño  $n \times 1$

Expresamos el riesgo empírico en notación matricial:

$$R(w) = (y - w)^T (y - w)$$

Gradiente:

$$\nabla_w R(w) = {}^T(y - w) = {}^T y - {}^T w$$

Minimizamos el riesgo empírico:

$$\nabla_w R(w) = 0 \Rightarrow w^* = ({}^T)^{-1} {}^T y$$

Recuperamos mínimos cuadrados ordinarios!

Posibles problemas del modelo:

- Teóricos:
  1. asumimos que  $y$  depende linealmente de  $x$
  2. asumimos que el modelo está especificado correctamente (no faltan variables)
- Numéricos:
  1. hay menos variables que observaciones
  2. no hay dos variables con correlación perfecta

# Selección de modelos

- Para medir la calidad del modelo, podemos calcular el riesgo o error empírico en el conjunto de entrenamiento
- Este error se puede disminuir de forma casi arbitraria aumentando la complejidad de la clase de funciones
- **Ejemplo:** en el caso de la regresión lineal, podemos añadir nuevas variables que sean expansiones polinómicas de las ya existentes
- Interesa el **error de generalización**, es decir, el error en nuevas observaciones no usadas para entrenar el modelo
- Partir los datos iniciales en dos conjuntos:
  1. conjunto de entrenamiento
  2. conjunto de test

# Equilibrio sesgo-varianza

- Asumimos que los datos han sido generados por

$$Y = f(X) + \epsilon$$

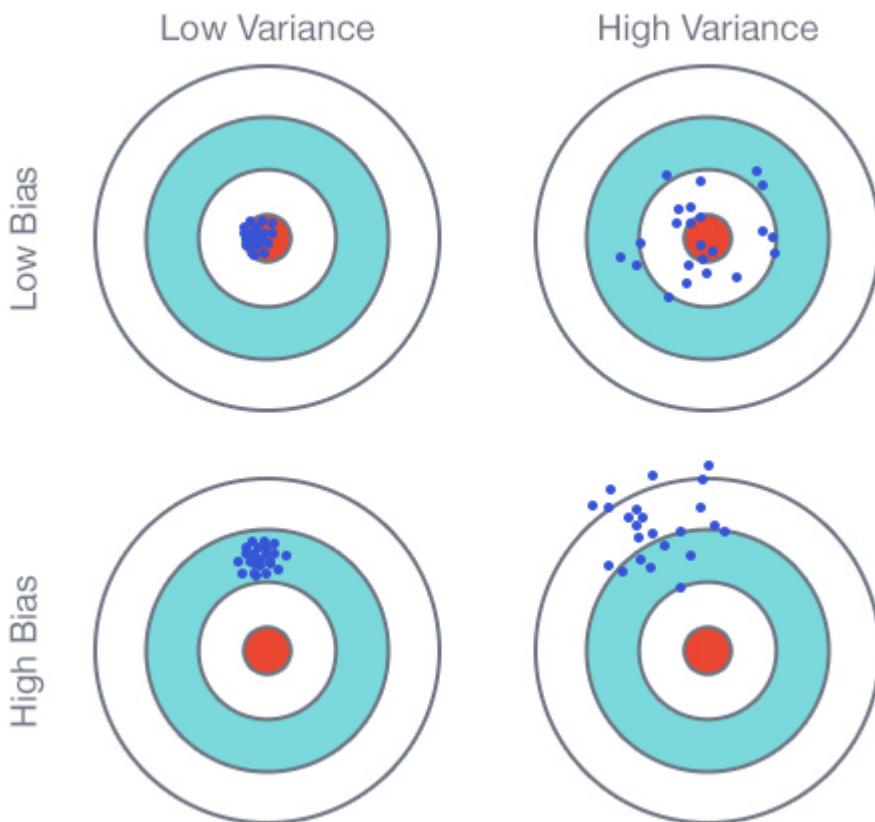
con  $[\epsilon] = 0$  y  $\text{Var}(\epsilon) = \sigma^2$

- El error esperado de un estimador  $\hat{f}(X)$  en el punto  $x$  (usando pérdida cuadrática) es

$$\text{EPE} = [(Y - \hat{f}(x))^2]$$

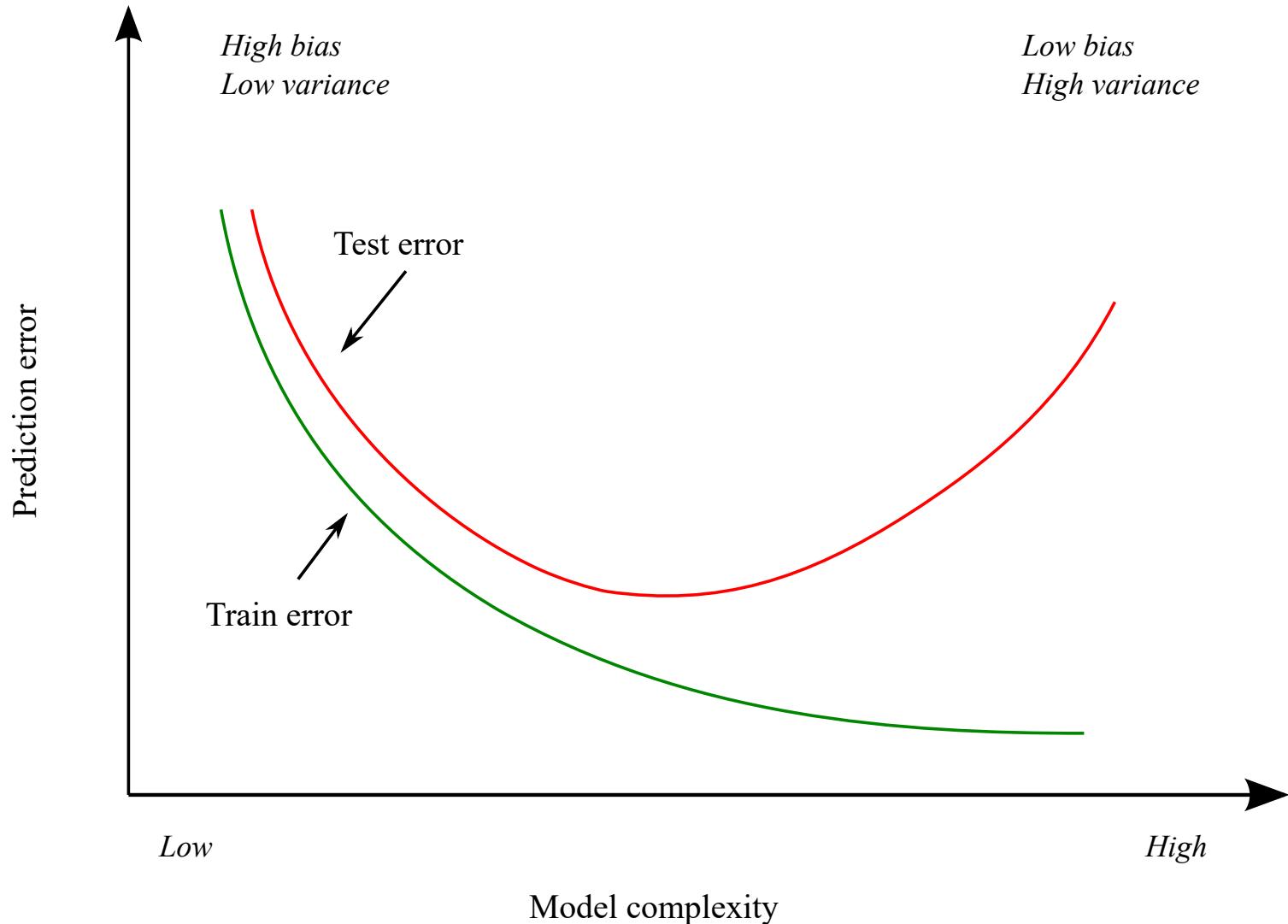
- Podemos descomponerlo en:

$$\text{EPE} = \underbrace{\left( [\hat{f}(x)] - f(x) \right)^2}_{\text{Sesgo}^2} + \underbrace{E\left[ \hat{f}(x) - [\hat{f}(x)] \right]^2}_{\text{Varianza}} + \underbrace{\sigma^2}_{\text{Ruido}}$$



# Sobreajuste

- Los términos de sesgo y varianza son opuestos: si disminuimos uno aumenta el otro y viceversa
- El término de ruido es inherente a los datos
- Si el modelo es muy simple, el estimador está sesgado y no se ajusta bien a los datos (infraajuste)
- Si el modelo es demasiado complejo, es muy sensible a pequeñas variaciones en los datos
- Además, el error de test será mucho más alto que el error de entrenamiento (**sobreajuste**)
- **Solución:** encontrar un equilibrio que minimice el error en el conjunto de test



# Simulación

```
set.seed(1)
n <- 10
x <- seq(0, 1, length.out = n)
y <- 1.5*x - x^2 + rnorm(n, 0, 0.05)
data <- data.frame(x=x, y=y)

x_new <- seq(0, 1, length.out=500)
newdata <- data.frame(x=x_new)

fit1 <- lm(y ~ x + I(x^2), data=data)
fit2 <- lm(y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5) +
           + I(x^6) + I(x^7) + I(x^8) + I(x^9),
           data=data)
fit3 <- lm(y ~ x, data=data)

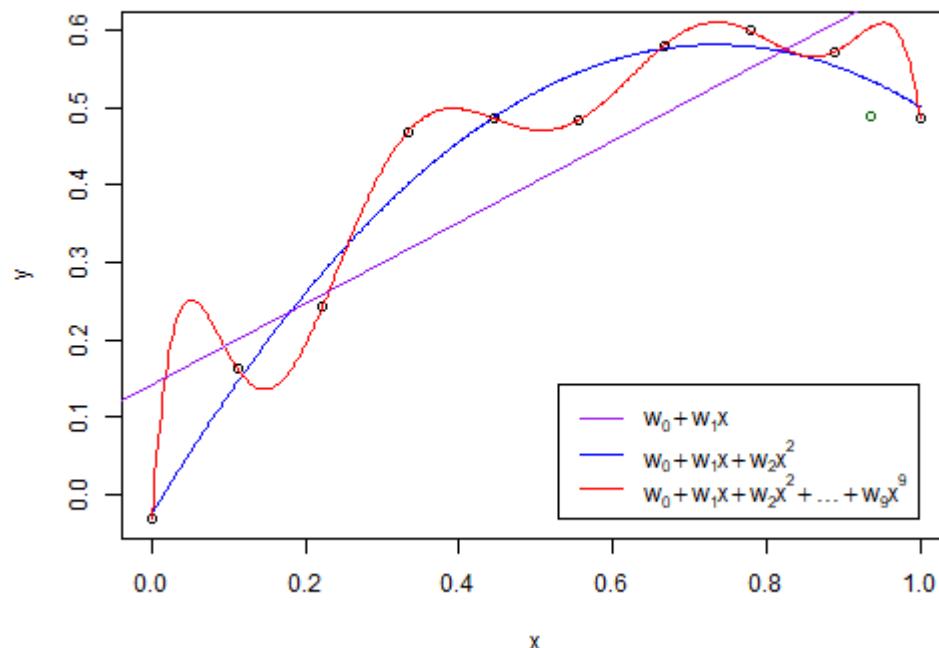
y_pred1 <- predict(fit1, newdata=newdata)
y_pred2 <- predict(fit2, newdata=newdata)

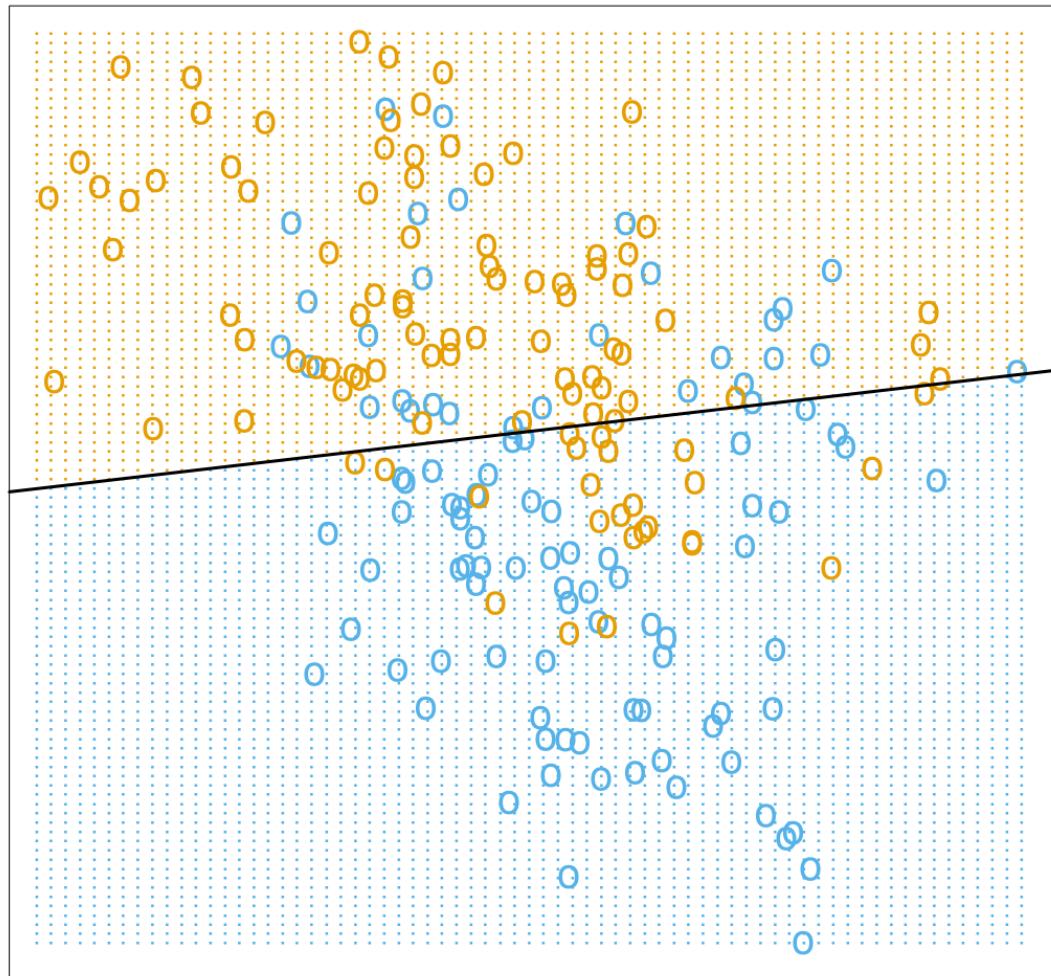
ntest <- 1
xtest <- runif(ntest)
ytest <- 1.5*xtest - xtest^2 + rnorm(ntest, 0, 0.05)
```

```

plot(data)
lines(x_new, y_pred1, col="blue")
lines(x_new, y_pred2, col="red")
abline(fit3, col="purple")
points(xtest, ytest, col="darkgreen")
legend("bottomright",
       c(expression(w[0] + w[1]*x),
         expression(w[0] + w[1]*x + w[2]*x^2),
         expression(w[0] + w[1]*x + w[2]*x^2 + ldots + w[9]*x^9)),
       lty=1, lwd=1.5, col=c("purple", "blue", "red"), inset=0.04)

```





Ejemplo de clasificación en 2 dimensiones [ESL]

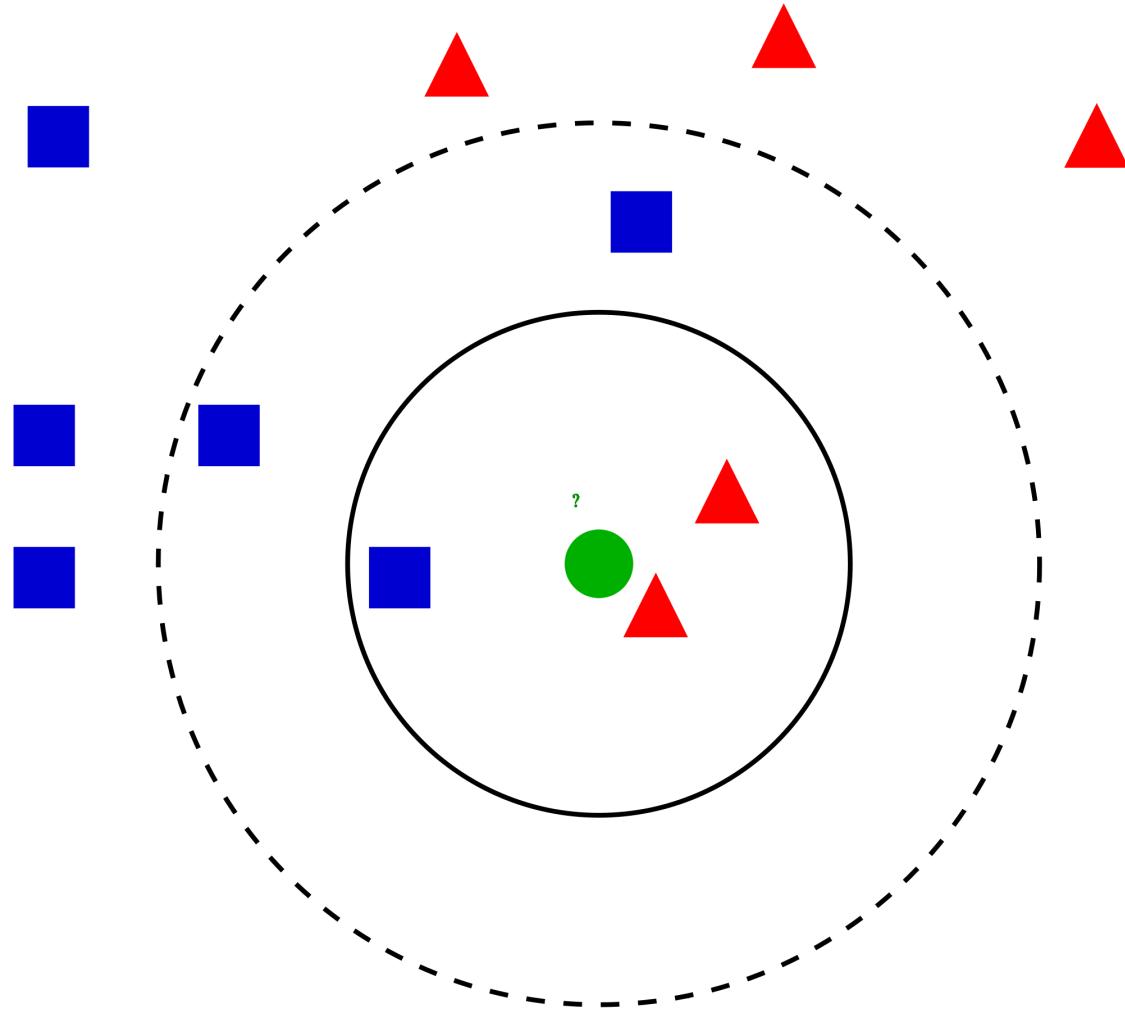
# Vecinos próximos

- Modelo sencillo que usa las observaciones cercanas a  $x$  para realizar la predicción:

$$f(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i$$

donde  $N_k(x)$  son las  $k$  observaciones más cercanas

- Necesaria una métrica (por ej. distancia euclídea)
- Se puede usar tanto para problemas de clasificación como regresión
- Muy sensible al valor de  $k$



```
library(class)

n <- nrow(iris)

# muestreo aleatorio
idx <- sample(n, n*0.75)

# partir en conjuntos de entrenamiento y test
train <- iris[idx, ]
test <- iris[-idx, ]

# separar variables indenpendientes de la clase (variable respuesta)
# entrenamiento
y_train <- train[, 5]
X_train <- train[, -5]

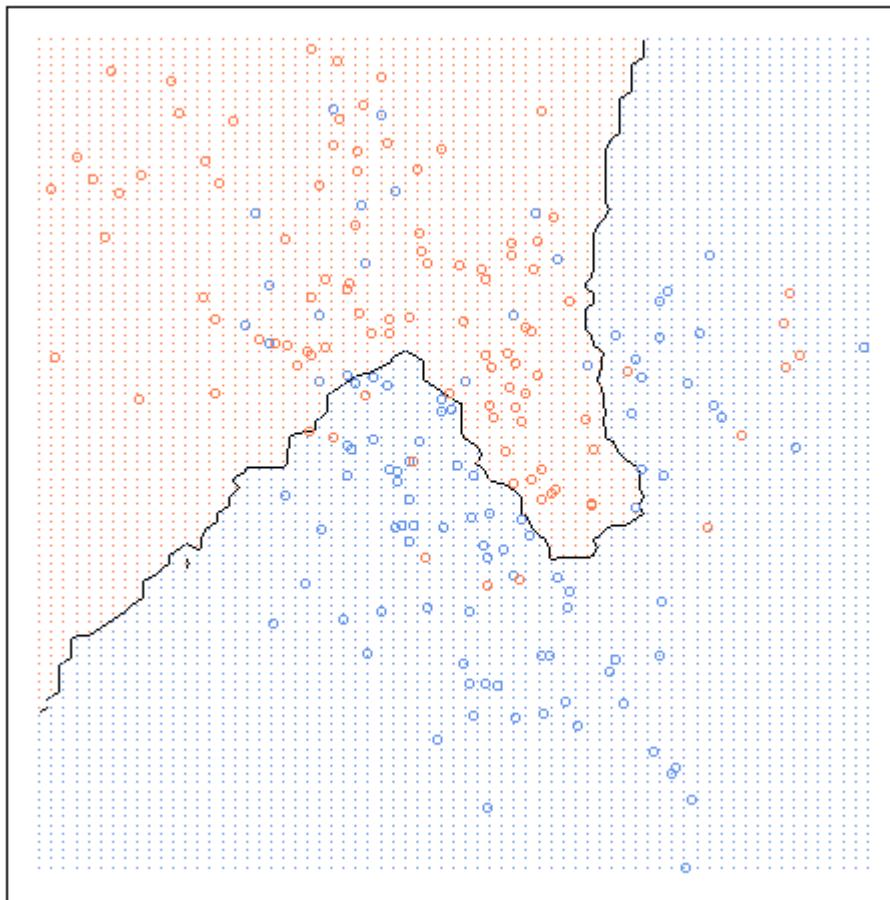
# test
y_test <- test[, 5]
X_test <- test[, -5]

# modelo knn
y_pred <- knn(X_train, X_test, y_train, k=3)

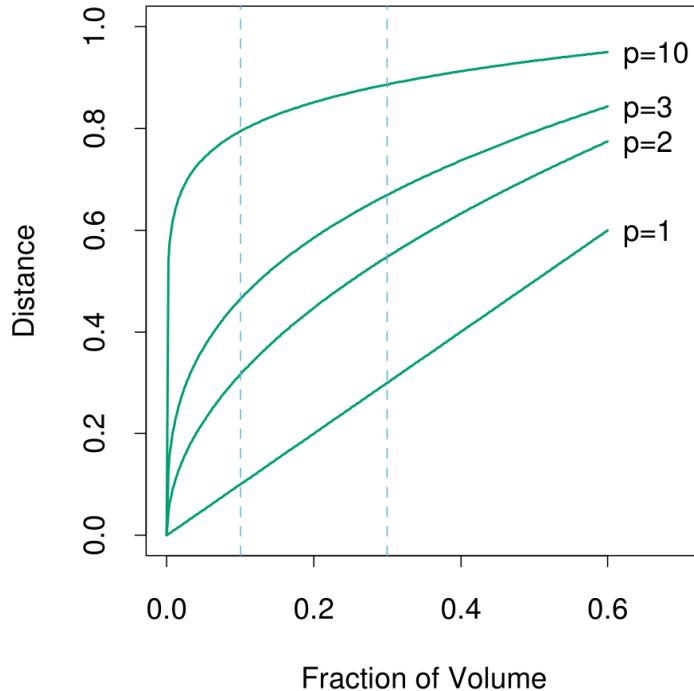
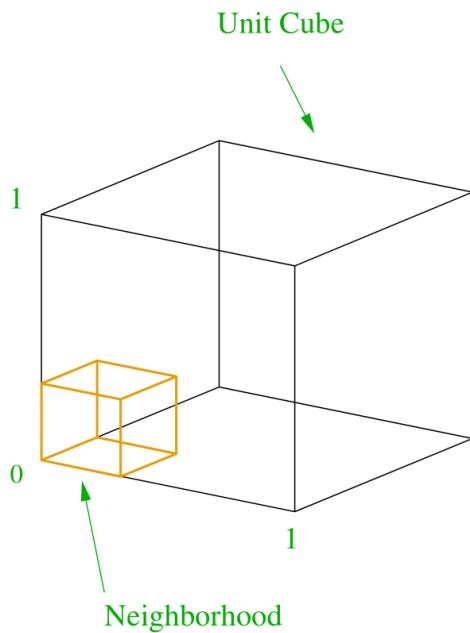
# tasa acierto
mean(y_test == y_pred)*100
```

```
## [1] 97.36842
```

**Vecinos próximos, k=15**



# Maldición de la dimensionalidad



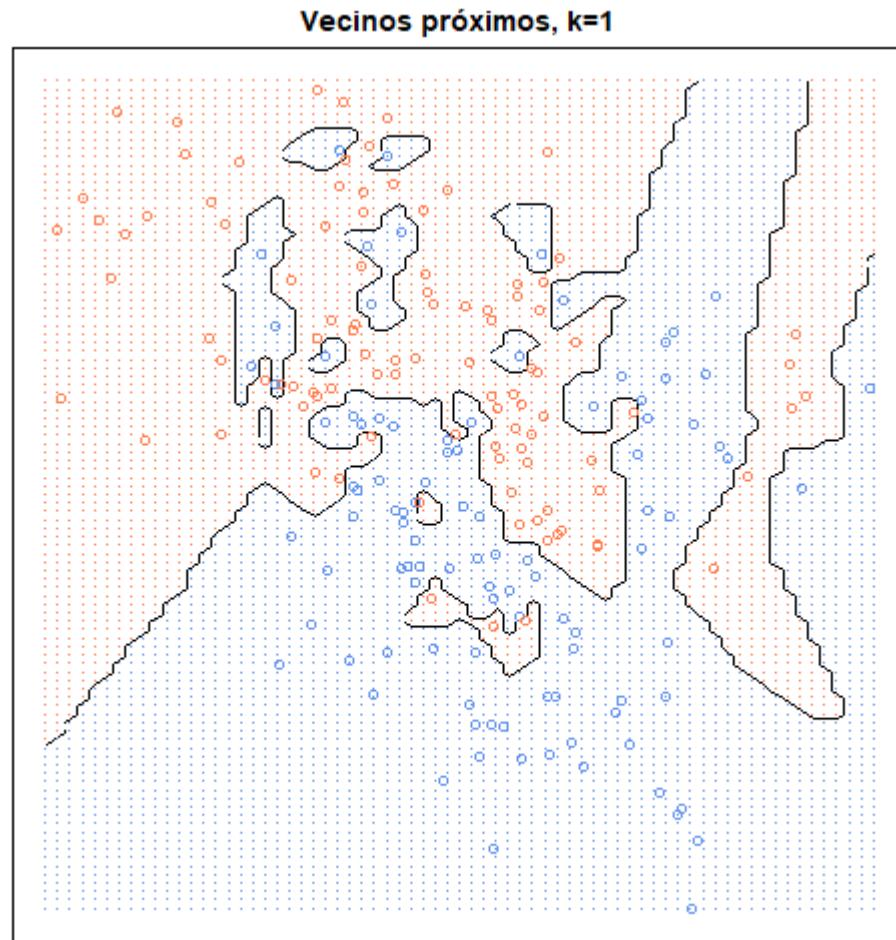
**FIGURE 2.6.** The curse of dimensionality is well illustrated by a subcubical neighborhood for uniform data in a unit cube. The figure on the right shows the side-length of the subcube needed to capture a fraction  $r$  of the volume of the data, for different dimensions  $p$ . In ten dimensions we need to cover 80% of the range of each coordinate to capture 10% of the data.

- En 2 dimensiones, para capturar un 10% del volumen necesitamos cubrir aprox. el 25% del rango de cada coordenada
- En 10 dimensiones, necesitamos el 80%
- Otro problema relacionado es la densidad del muestreo:
  - Si en 1 dimensión necesitamos  $n = 100$  muestras para cubrir el espacio, en 10 dimensiones necesitamos  $n = 100^{10}$  para tener la misma densidad de muestreo (crecimiento exponencial)
- A medida que la dimensión crece, las observaciones están mas cerca de las esquinas del hipercubo que del centro
  - Observaciones cercanas a las esquinas son más difíciles de clasificar porque el valor de sus atributos cambia muy bruscamente
- Resumiendo, si  $d$  es muy grande, métodos como KNN basados en distancias tienen problemas

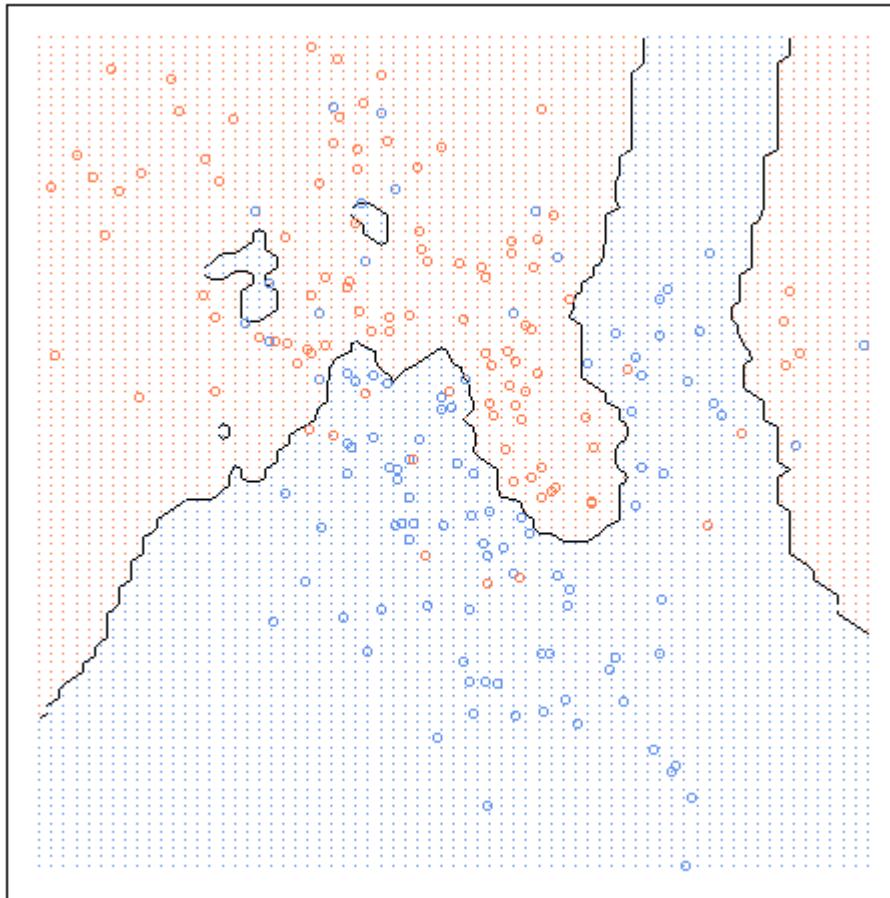
# Regresión lineal vs vecinos próximos

- La frontera de decisión de la regresión lineal es suave: tiene poca varianza pero potencialmente mucho sesgo
- $k$ -vecinos próximos no asume ninguna estructura en los datos:
  - la frontera de decisión depende localmente solo de los  $k$  puntos más cercanos
  - tiene poco sesgo pero mucha varianza, ya que es muy inestable
- Elegir un modelo u otro depende de los datos del problema

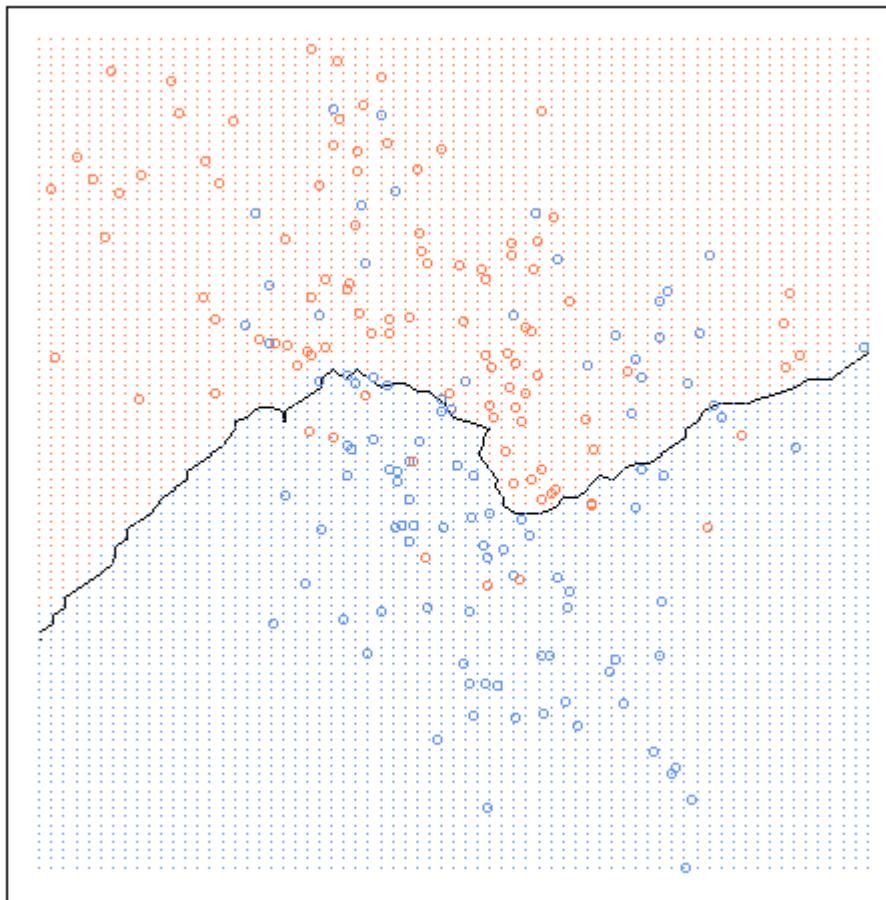
# Vecinos próximos: dependencia de $k$



**Vecinos próximos, k=5**



**Vecinos próximos, k=50**



# Selección de hiper-parámetros

- $k$  es un **hiper-parámetro** que controla la complejidad del modelo
- Podemos realizar un argumento similar a la comparación con la regresión lineal:
  - Para  $k$  grande, la frontera es más suave pero tiene (potencialmente) mayor sesgo
  - Para  $k$  pequeño la frontera es muy inestable (mayor varianza), pero menos sesgo
- Nota: usar el error de entrenamiento para elegir el valor de  $k$  es mala idea, para  $k = 1$  tenemos error 0!!
- Los distintos valores de  $k$  se pueden comparar usando el conjunto de test

# Conjunto de validación

- Elegir  $k$  como el valor que minimiza error de test → error de test ya **no** es una buena estimación del rendimiento del modelo en nuevos datos
- Lo mismo ocurre si elegimos la clase de funciones (modelo) usando el error de test
- **Solución:** crear un tercer conjunto, conjunto de validación, para seleccionar hiperparámetros y comparar modelos
- Finalmente, reportar el error de test como estimación del poder de generalización del modelo

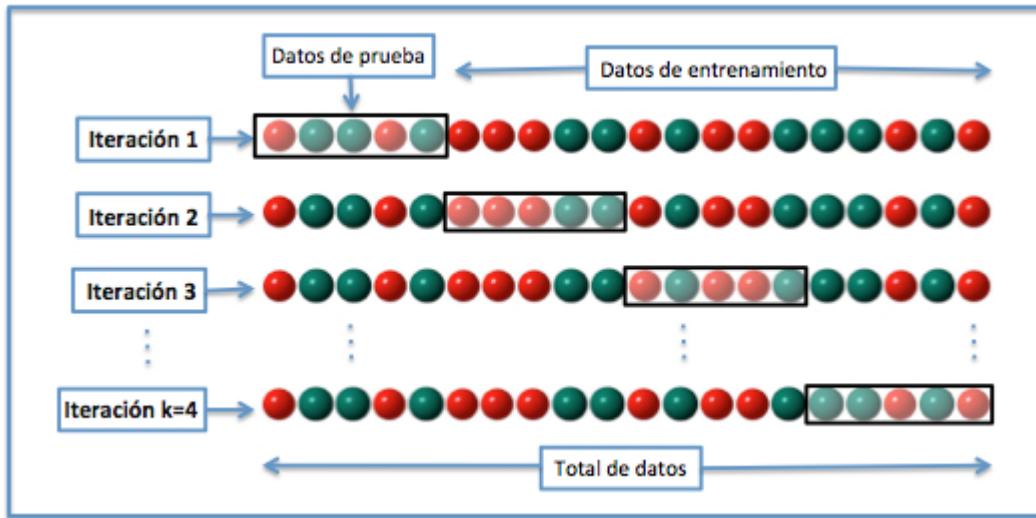
# Validación cruzada

- Se divide el conjunto de entrenamiento en  $K$  particiones
- Usar  $K - 1$  particiones como entrenamiento y la otra como test para ajustar  $K$  modelos
- $\hat{f}^{-k}(x)$  es el modelo entrenado con todas las particiones menos la  $k$
- Calcular el error de validación cruzada:

$$\text{CV}(\hat{f}) = \frac{1}{n} \sum_{i=1}^n L(y_i, \hat{f}^{-\kappa(i)}(x_i))$$

donde  $\kappa : \{1, \dots, n\} \rightarrow \{1, \dots, K\}$  es una función que indica a qué partición pertenece cada observación  $i$

- Cuando  $K = n$  se conoce como validación cruzada *leave-one-out*



Wikipedia. Cross-validation)

# Regularización

- A menudo se puede reducir la varianza de un estimador a cambio de introducir un pequeño sesgo
- Este término también puede inducir propiedades en la solución, por ej. *sparsity*
- Para ello limitamos la complejidad del modelo añadiendo a la función de pérdida un término de **regularización**

$$\hat{f} = \arg \min_f \{L(y, f(x)) + \lambda J(f)\}$$

- Muchos modelos en aprendizaje automático encajan en este paradigma

# Ejemplo

- El estimador de mínimos cuadrados es el mejor estimador no sesgado (mejor = menos varianza)
- Un término de regularización muy habitual es la norma  $l_2$ :

$$\|w\|_2^2 = w^T w$$

- Junto con la función de pérdida de la regresión lineal, el modelose conoce como regresión ridge:

$$w^* = \arg \min_w \{(y - w)^T (y - w) + \lambda w^T w\}$$

- Tomando derivadas e igualando a 0 la solución es

$$w^* = (\mathbb{I}^T + \lambda)^{-1}(\mathbb{I}^T y)$$

donde  $\mathbb{I}$  es la matriz identidad

# Regresión ridge en R

- La función `lm.ridge()` de la librería MASS entrena una regresión lineal con regularización *ridge* para varios valores del parámetro  $\lambda$
- La librería `ridge` implementa la función `linearRidge()` que selecciona automáticamente el valor óptimo del parámetro  $\lambda$  usando el método propuesto en [Cule et al \(2012\)](#)

# Regresión logística

Equivalente a la regresión lineal para problemas de clasificación

**Objetivo:** estimar la probabilidad a posteriori de pertenencia a cada clase

Función logística o sigmoidea:

$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$

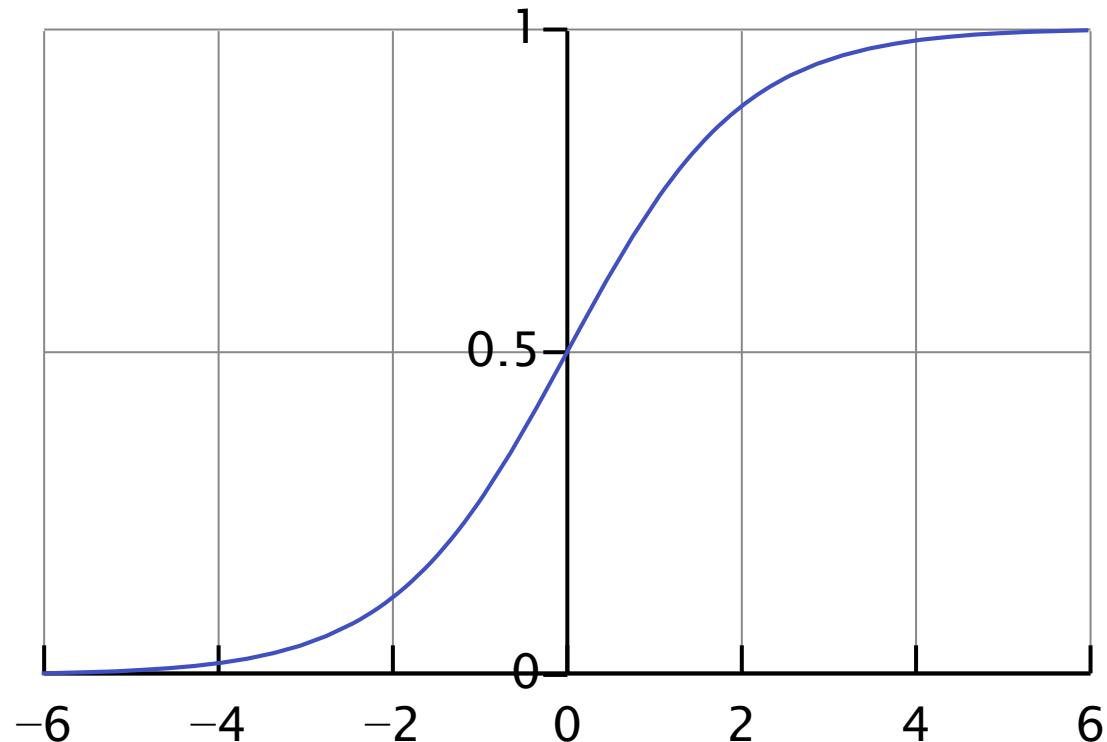
Para 2 clases con etiquetas  $\{0, 1\}$ , definimos:

$$p(y_i = 1 | x_i) = \sigma(w^T x_i)$$

Por tanto

$$p(y_i = 0 | x_i) = 1 - \sigma(w^T x_i) = \sigma(-w^T x_i)$$

# Curva logística



Wikipedia, [Logistic function](#)

# Interpretación: odds ratio

- Dado un modelo lineal con una única variable independiente, el **odds ratio** se define como

$$\text{OR} = \frac{\exp(w_0 + w_1(x + 1))}{\exp(w_0 + w_1x)} = \exp(w_1)$$

- Es decir, como cambian las probabilidades de la salida cuando la variable  $x$  aumenta una unidad:
  1. Si  $\text{OR} = 1$  la variable  $x$  no tiene ninguna asociación con la salida
  2. Si  $\text{OR} > 1$  la variable está asociada con una mayor probabilidad de la salida
  3. Si  $\text{OR} < 1$  la variable está asociada con una menor probabilidad de la salida

# Calidad modelos clasificación

		Predicted class	
		<i>P</i>	<i>N</i>
<i>P</i>	<i>P</i>	True Positives (TP)	False Negatives (FN)
	<i>N</i>	False Positives (FP)	True Negatives (TN)

Medidas:

- Tasa de acierto:  $\frac{TP+TN}{P+N}$
- Sensitividad, recall, TPR:  $\frac{TP}{TP+FN}$
- Especificidad, TNR:  $\frac{TN}{TN+FP}$
- Precisión, PPV:  $\frac{TP}{TP+FP}$
- F1-score:  $2 \times \frac{PPV \times TPR}{PPV + TPR}$

# Regresión logística: optimización

Estos modelos se optimizan usando el log de la verosimilitud condicionada a  $\mathbf{x}$  :

$$L(\mathbf{w}) = \sum_{i=1}^n y_i \log[\sigma(\mathbf{w}^T \mathbf{x}_i)] + (1 - y_i) \log[\sigma(-\mathbf{w}^T \mathbf{x}_i)] = y_i(\mathbf{w}^T \mathbf{x}_i) + \log[\sigma(-\mathbf{w}^T \mathbf{x}_i)]$$

Derivada:

$$\partial_{\mathbf{w}} L(\mathbf{w}) = \mathbf{x}_i(y_i - \sigma(\mathbf{w}^T \mathbf{x}_i))$$

Son  $d + 1$  ecuaciones no lineales en  $\mathbf{w}$

Se pueden resolver usando una variante específica de Newton-Raphson: Iteratively Re-weighted Least Squares (IRLS)

Segunda derivada:

$$\partial^2 L(\mathbf{w}) = - \sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^T \sigma(\mathbf{w}^T \mathbf{x}_i)(1 - \sigma(\mathbf{w}^T \mathbf{x}_i))$$

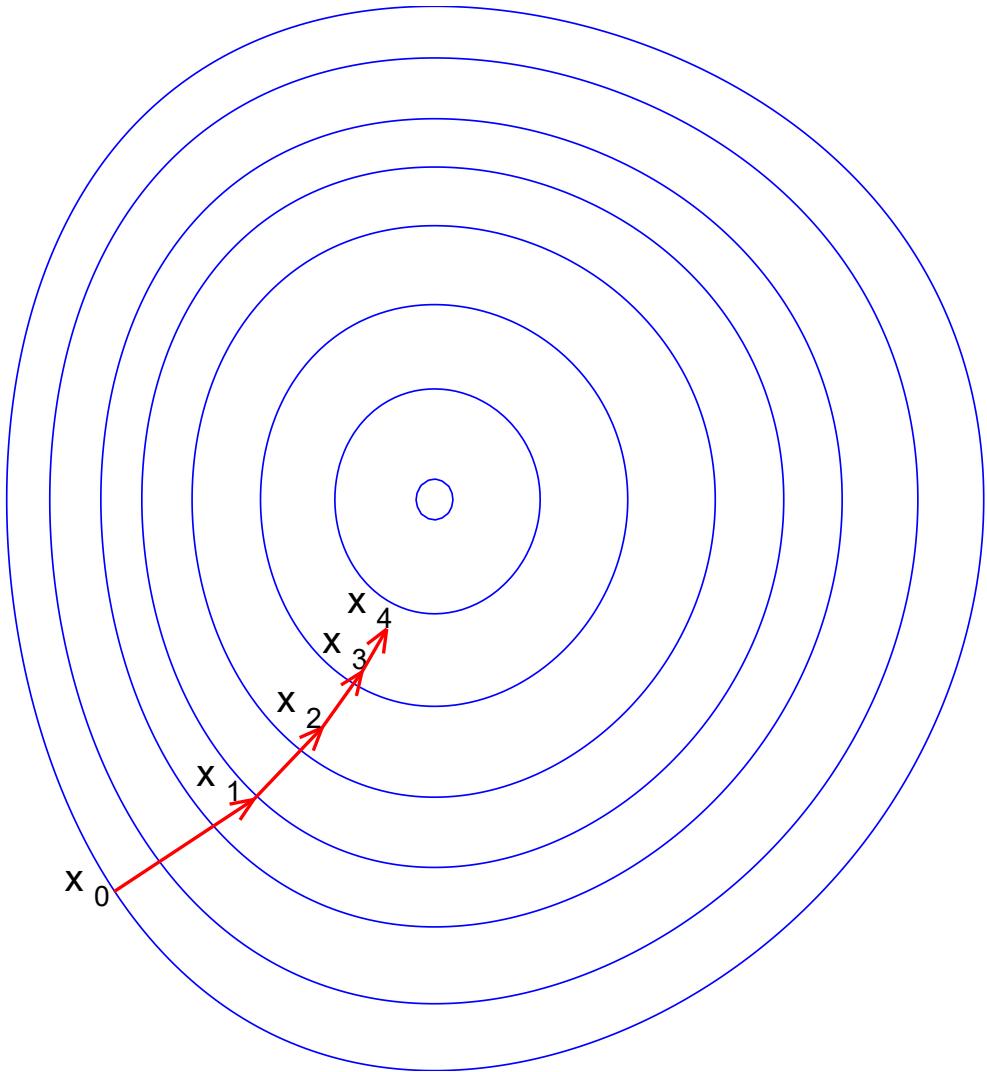
# Optimización numérica: Introducción

# Descenso por gradiente

- Dada una función convexa y diferenciable  $f(w)$ , podemos encontrar su mínimo iterando:

$$w^{k+1} = w^k - \eta \nabla f(w^k)$$

- $\eta$  es un parámetro que controla la longitud del paso
- Si  $\eta$  es suficientemente pequeño, el valor de  $f(w)$  decrece de forma monótona en cada paso
- También se puede demostrar que la secuencia  $\{w^k\}$  converge al óptimo  $w^*$
- Método de primer orden, ya que solo usa información de la primera derivada (gradiente)



# Newton-Raphson (N-R)

- Queremos minimizar una función convexa dos veces diferenciable,  $f : \mathbb{R}^d \rightarrow \mathbb{R}$
- Expansión de Taylor de segundo orden de  $f$  en  $x$ :

$$f(x + h) \approx f(x) + \nabla f(x)^T h + \frac{1}{2} h^T \mathbf{H} h$$

donde  $\mathbf{H}$  es la matriz de segundas derivadas (Hessiana)

- Entonces

$$\nabla_h f(x + h) = \nabla f(x) + \mathbf{H} h$$

- Por tanto el  $h$  que minimiza la aproximación de segundo orden es

$$h^* = -\mathbf{H}^{-1} \nabla f(x)$$

- Iteramos:

$$x^{k+1} = x^k - \mathbf{H}^{-1} \nabla f(x)$$

# N-R para regresión logística

Resscribimos las derivadas en notación matricial:

$$\begin{aligned}\partial \quad (w) &= {}^T(y - p) \\ \partial^2 \quad (w) &= - {}^T\end{aligned}$$

donde  $p_i = \sigma(w^T x_i)$  y  $\quad_{ii} = \sigma(w^T x_i)(1 - \sigma(w^T x_i))$

Actualización de N-R:

$$\begin{aligned}w^{k+1} &= w^k + ({}^T) {}^{-1} {}^T(y - p) \\ &= ({}^T \quad {}^T) {}^T z\end{aligned}$$

con  $z = w^k + {}^{-1}(y - p)$

Es decir, en cada iteración se resuelve el problema de mínimos cuadrados

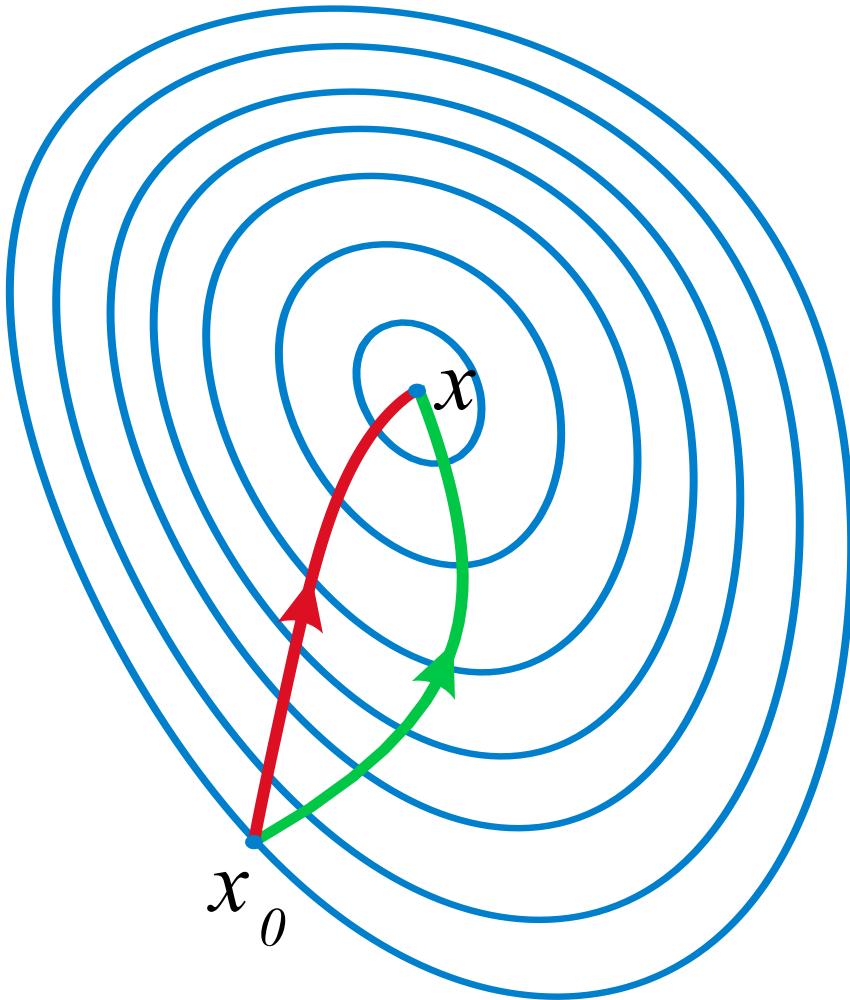
$$w^{k+1} = \arg \min_w (z - w)^T (z - w)$$

# *Iteratively Re-weighted Least Squares*

- Dado un valor inicial  $w^0$
- Iterar para  $k = 0, 1, 2, 3, \dots$ 
  1. Calcular  $p^k$  con  $p_i = \sigma((w^k)^T x_i)$
  2. Calcular  $\Sigma^k$  con elementos diagonales  $\Sigma_{ii}^k = \sigma((w^k)^T x_i)(1 - \sigma((w^k)^T x_i))$
  3. Calcular  $z^k = w^k + (\Sigma^k)^{-1}(y - p^k)$
  4. Actualizar pesos  $w$

$$w^{k+1} = \arg \min_{w^k} (z^k - w^k)^T \Sigma^k (z^k - w^k)$$

- Hay que establecer un criterio de parada
- Puede tener problemas de convergencia dependiendo del valor inicial



# Aprendizaje supervisado en la práctica

# Primeros pasos

- Los datos a analizar a menudo provienen de fuentes variadas (redes sociales, sensores, encuestas, ...) y están almacenados en diferentes soportes (ficheros de texto, base de datos, ficheros binarios, streams...)
- Lo primero es identificar el problema qué queremos resolver y cuales son las variables que tenemos disponibles y pueden aportar información
- Ante la duda, no descartar variables/información ni observaciones antes de tiempo
- Lo segundo es combinar todas esa información y transformarla en una mezcla de variables numéricas (valores continuos) y categóricas (valores discretos)
- El objetivo final del preproceso es organizar esos datos en un formato tabular (filas y columnas)

# Distintos tipos de información

- En ocasiones no es trivial transformar ciertos tipos de información en variables numéricas y/o categóricas
- Para estos casos a menudo es necesario un preproceso extra, muy dependiente del problema a resolver y específico del dominio
- Ejemplos:
  1. Texto (tweets, páginas web, documentos): word2vec, bag-of-words, modelos n-gram
  2. Imágenes: valores RGB de los píxeles, intensidad de gris
  3. Audio: transformada de Fourier, coeficientes MFCC
  4. Video: secuencia de frames
  5. Series temporales: añadir *lags* como variables

# Valores que faltan

- Es importante distinguir cuando una variable tiene valor 0 de "no conocido"
- Si es posible, nos gustaría saber también el mecanismo que origina el valor que falta:
  1. MCAR (*Missing Completely At Random*): el motivo por el que falta un valor no está relacionado con otras variables
  2. MAR (*Missing At Random*): el motivo está relacionado con otras variables
  3. NMAR (*Not Missing At Random*): no sabemos el motivo por el que falta
- Estos valores pueden venir representados por múltiples caracteres (“\*”, “-”, campo vacío, etc.)
- Hay que codificarlos de manera especial para tenerlos en cuenta en los análisis (en R NA )

**Ejemplo:** en datos que provienen de un reconocimiento médico varios pacientes no tienen ningún valor en el campo de “Fármacos”. ¿No toman ninguna medicación o el médico no ha registrado la respuesta?

# Completar valores que faltan

1. Ignorar observaciones donde falte alguna variable

2. Completar observaciones con:

- media o mediana, datos continuos
- moda, datos categóricos

usando:

- todas las observaciones de esa variable
- observaciones pero agrupadas por otras variables

3. Modelos predictivos

- Paquete `mice` en R
- KNN: usar solo observaciones cercanas

Otras **librerías**

# Valores extremos

- Distinguir si un valor es erróneo o válido pero extremo es muy complicado y dependiente del dominio
- Existen diversas reglas para identificarlos (paquete `outliers`)
- Pueden perjudicar a ciertos algoritmos de aprendizaje, mientras que otros son robustos frente a este tipo de datos
- Comprobar siempre que no hay valores imposibles

Ejemplo: en datos provenientes de un reconocimiento médico, aparece un paciente con un IMC de 50

# Tratamiento valores extremos

1. Eliminar la observación
2. Asignar como nuevo valor el límite inferior/superior de los valores normales
3. Asignar al valor extremo NA e imputar su nuevo valor usando las técnicas para valores que faltan

# Normalización

- Las variables numéricas suelen tener rangos muy diversos
- **Ejemplo:** salario (10,000 – 100,000 EUR) y edad (0–100)
- Algunos modelos interpretan esta diferencia de escalas como que unas variables son más importantes que otras o las observaciones están más próximas
- En ocasiones normalizar las variables también puede ayudar a que el algoritmo de optimización converja más rápido
- Cuidado al analizar los resultados, ya que están en los nuevos rangos

# Métodos

- Media 0 varianza 1 (estandarizar):

$$z = \frac{x - \text{mean}(x)}{\text{std}(x)}$$

- Escalar a un intervalo, por ej.  $[-1, 1]$ :

$$z = \frac{x - \min(x)}{\max(x) - \min(x)}$$

- Estandarización robusta:

$$z = \frac{x - \text{median}(x)}{\text{IQR}(x)}$$

- Otros

# Variables categóricas

- Muy comunes en todo tipo de fuentes de datos
- Muy pocos algoritmos de aprendizaje son capaces de tratarlas directamente
- Por tanto, tenemos que convertirlas en numéricas
- La transformación donde se asigna a cada uno de sus valores un número entero no suele ser buena idea, ya que crea una relación artificial de orden y falsea las distancias
- Estándar: utilizar una codificación *dummy* (*one-hot encoding*)

# Codificación *dummy*

Edad	Sexo		Edad	Es mujer?	Es hombre?
34	H		34	0	1
18	M	⇒	18	1	0
67	M		67	1	0
21	M		21	1	0
15	H		15	0	1

- Finalmente, podemos eliminar una de las dos nuevas variables puesto que tienen correlación 1
- En general, para una variable categórica con  $p$  valores añadimos  $p - 1$  variables nuevas

# Otras codificaciones

- Codificación **ordinal**: variables ordinales, por ej. puntuación:  $\{\text{baja, media, alta}\} \Rightarrow \{1,2,3\}$  (cuidado con las distancias!)
- Codificación **binaria**: variables ordinales con alta dimensión, transformar la codificación a código binario
- **Hashing**: codificar los valores usando una función de *hash*, que transforma cualquier valor a un número fijo de bits (columnas)
- **Leave One Out**: el valor  $C$  de la variable categórica para la observación  $i$  se codifica como la media de la salida para todas las observaciones con valor  $C$  en la variable categórica, excluyendo  $i$
- **Codificación de similitud**
- ...

# Codificaciones específicas del dominio

- Ejemplo si hay relación de orden y no queremos falsear las distancias:

Mes	Día	Temp.		Días desde 01/01	Temp.
Enero	29	22.2		29	22.2
Enero	30	27.8	⇒	30	27.8
Enero	31	28.6		31	28.6
Febrero	1	26.1		32	26.1
Febrero	2	25.3		33	25.3

# Variables categóricas en R

- Las variables categóricas en R se codifican con el tipo `factor`
- Muchas funciones que implementan algoritmos de aprendizaje con interfaz para fórmulas aceptan factores directamente y los convierte usando una codificación *dummy*
- Tenemos que hacer la codificación explícitamente si:
  1. el algoritmo no tiene interfaz para fórmulas y solo acepta variables numéricas
  2. queremos usar otra codificación distinta de la codificación *dummy*
- Función `dummy_cols()` de la librería `fastDummies`
- Librería `vtreat`