# Remuestreo y CV

Matias Bajac

2024-09-22

## Remuestreo

- Hemos mencionado que para evaluar los modelos con distintos numeros de parametro es importante separar la muestra en entrenamiento y testeo
- **NO usar** los mismos datos para entrenar el modelo y evaluarlo
- Estimar el error de prediccion con los datos de entrenamiento subestima el error
- El error test se puede calcular facilmente si contmos con un conjunto de testeo pero hay veces que no contamos con el.
- En ausencia de un conjunto de test que pueda ser usado para estimar el error de testeo hay un conjunto de tecnicas que permiten estimar el mismo usando el conjunto de entrenamiento disponible.
- Algunas tecnicas ajustan el error de entrenamiento para estimar el error de testeo

## Tecnicas de remuestreo

- Se estima el error de testeo manteniendo afuera un conjunto de observaciones de entrenamiento.
- **Idea general**: se extraen **sucesivas muestras** del conjunto de **entrenamiento**, se ajusta el modelo en cada muestra para obtener informacion adiional sobre el modelo ajustado.

## Aplicaciones habituales

En general:

- Generar(re) - muestras a partir de UN conjunto de datos de entrenamiento.
- Se usan para evaluar propiedades estadisticas: error estandard, sesgo, error de prediccion.
- Seleccion de parametros de ajuste basados en re muestreo.

Dos grandes herramientas, Cross - Validation y Bootstrap.

- Estas tecnicas son tan generales que pueden ser usadas para la mmayoria de los metodos de aprendizaje

## Aplicacion: Error Validacion

- Nos gustaria estimar la **variabilidad** en el ajuste de un modelo de regresion lineal.
- Podemos usar alguna de las tecnicas de remuestreo para sacar **repetidas muestras** del conjunto de entrenamiento, ajustar un modelo de regresion para cada nuevo conjunto y luego examinar hasta que punto el ajusto difiere.
- Este procedimiento nos puede dar informacion que no tenemos si ajustamos el modelo solamente con los datos observados.

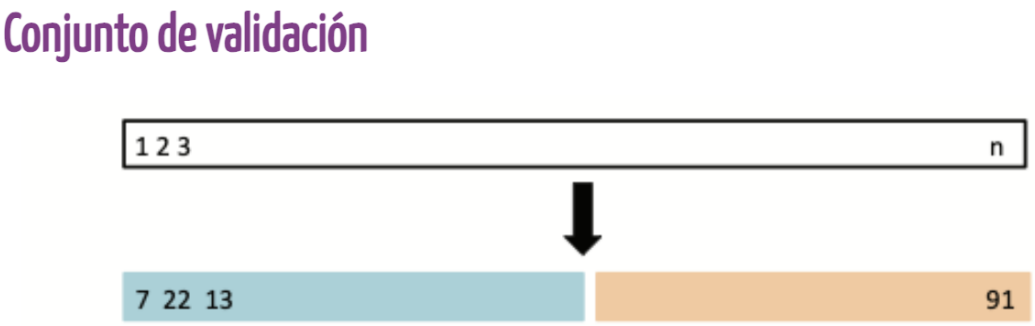
- Los metodos de remuestreo pueden tener un alto costo computacional ya que estamos ajustando muchas veces el modelo, pero actualmente el desarrollo computacional no es una limitante.

## Conjunto de validacion

-Queremos estmiar el error de prediccion, una aproximacion valida seria dividir aleatoriamente los datos en 2 sub conjuntos de similar tamanio (entrenamiento y validacion)

-El modelo se ajusta (fit) con los datos de entrenamiento y el modelo se usa para predecir en el conjunto de validacion.

Se calcula el **MSE** (regresion) que estima el **error de prediccion**




## Ejemplo. . .

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.0     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

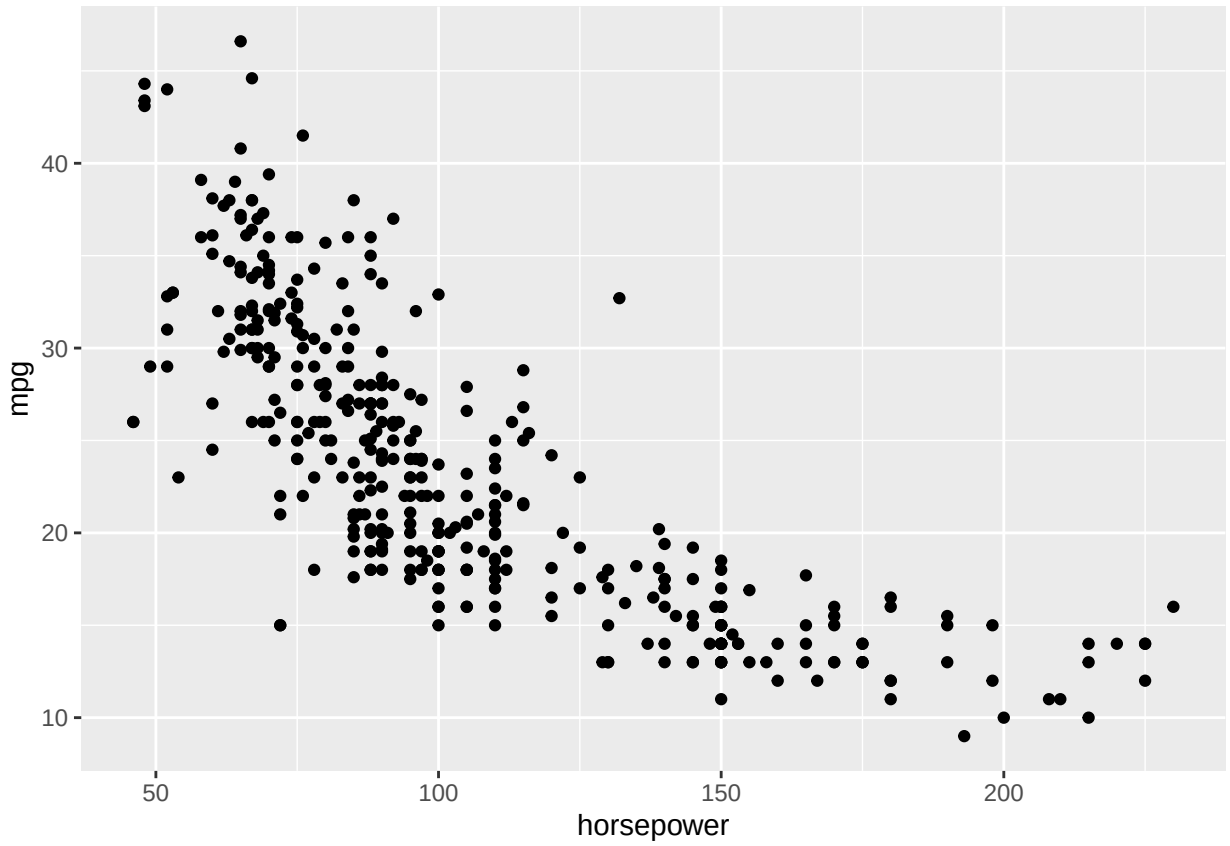
```
library(ISLR2)

Auto |> select(mpg, horsepower, year, name) |> head()
```

```
##   mpg horsepower year                       name
## 1  18        130   70 chevrolet chevelle malibu
## 2  15        165   70         buick skylark 320
## 3  18        150   70        plymouth satellite
```

```
## 4  16         150   70             amc rebel sst
## 5  17         140   70               ford torino
## 6  15         198   70          ford galaxie 500
```

Relacion gasto y potencia

```
ggplot(Auto) + geom_point(aes(x=horsepower, y=mpg))
```




## Ajuste modelo lineal

Estimacion del error en el conjunto de validacion

```
# Sorteamos indices para conjunto entrenamiento
set.seed(432)
tr <- sample(1:392, 196)

# Ajustamos modelo usando datos tr
fit <- lm(mpg ~ horsepower, subset = tr,  data = Auto)

# Calculamos error MSE en los datos NO utilizados
mean( (Auto$mpg - predict(fit, Auto))[-tr]^2 )
```

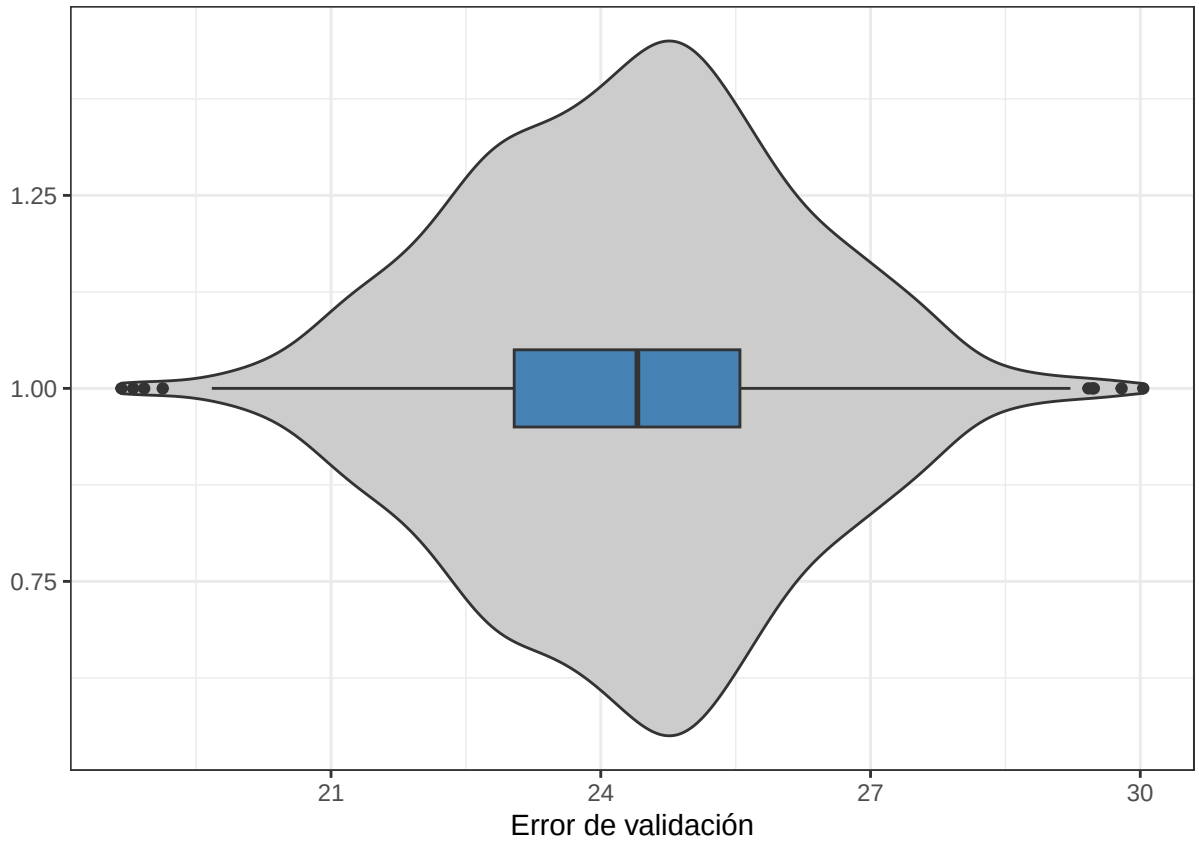
```
## [1] 23.7725
```

Repetimos el procedimiento R = 1000 veces

```
R <- 1000
err.val <- numeric(R)
```

```r
for (r in 1:R) {
  tr <- sample(1:392, 196)
  fit <- lm(mpg ~ horsepower, data = Auto, subset = tr)
  err.val[r] <- mean( (Auto$mpg - predict(fit, Auto))[-tr]^2 )
  }
```

```r
ggplot( mapping=aes(x=err.val, y=1) ) +
  geom_violin( fill='grey80'  ) +
  geom_boxplot(width=0.1, fill='steelblue') +
  labs(x='Error de validación', y='') +
  theme_bw()
```




## Conjunto de validacion

Simple de entender e implementar pero tiene 2 problemas

- La estimacion del error de prediccion puede ser muy variable dependiendo las observaciones seleccionadas en cada sub conjunto.
- Esta aproximacion tiende a sobrestimar el error de prediccion.

Veremos el metodo de validacion cruzada que intenta superar estos problemas

## Validacion cruzada

Validacion cruzada implica un refinamiento del conjunto de validacion que ataca sus dos problemas.

Metodo para estimar error de prediccion de un modelo.

- Comparar modelos, seleccionar el de menor error. **Evaluacion de modelos**
- Seleccionar **tunning parameters (Seleccion de modelos)**

##Error de prediccion##

Idea basica: Dividir los datos en subconjuntos, estimo el mdoelo con una parte (trenning ) y calculo el error prediciendo datos NO utilizados en la estimacion del modelo( test set)

Como dividir los datos? Trade off entre sesgo y varianza

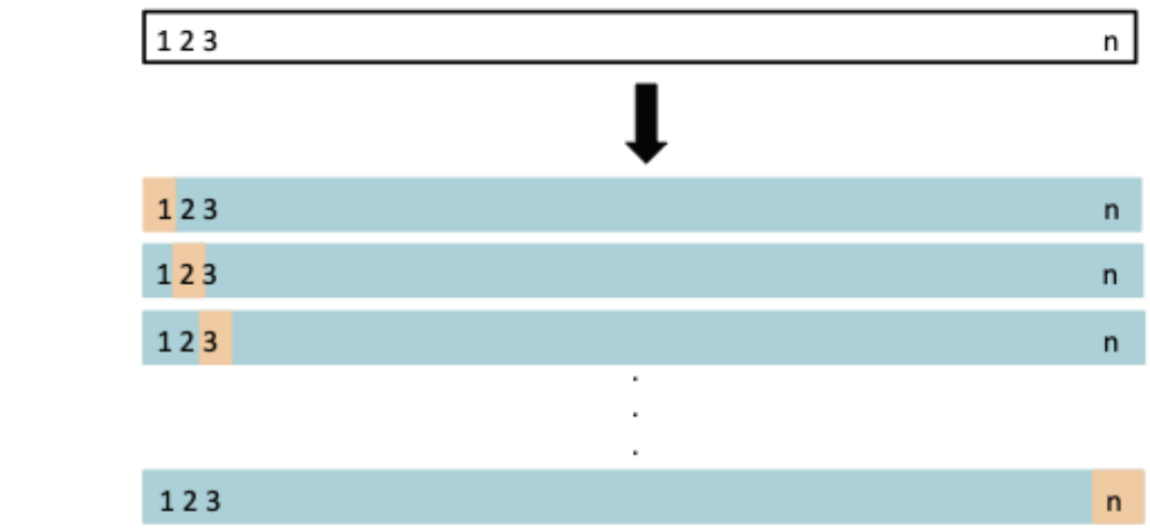
$$E[(Y - f(x))^2)] \ = \ \sigma^2 \ + (E[\hat{f}(x)] \ - f(x))^2 + E[\hat{f}(X) - E\hat{f}(X))^2]$$

Muchos datos para estimar pocos para predecir , **poco sesgo y mucha varianza**

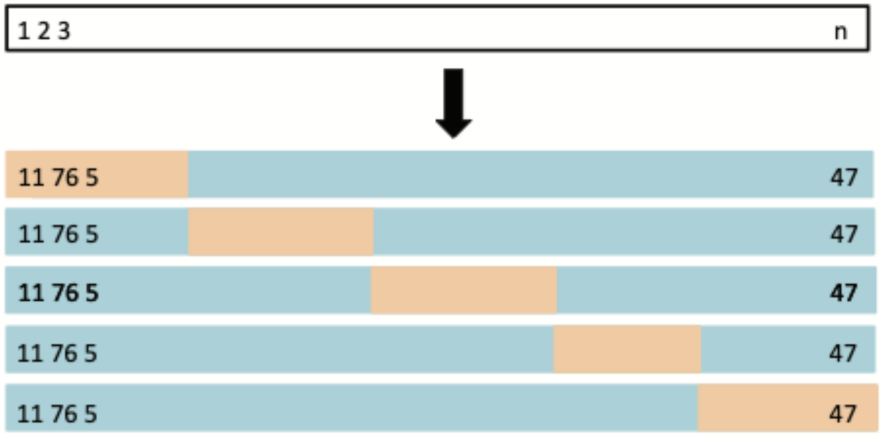
### Validacion cruzada

Varias formas de dividir nuestros conjuntos de datos, si tenemos n observaciones

- n -1 observaciones para estimar el modelo y 1 para predecir y repito esto n veces (Leave one out)
- k folds (grupos) con k<n: estimo con n-n/k y se predice con n/k, repito el procedimiento k veces.



## Validación cruzada: k < n subconjuntos




### Validacion cruzada

$$CV_k\hat{f} = \frac{1}{k}\sum_{j=1}^{k} MSE_j = \frac{1}{n}\sum_{i=1}^{n}(y_i - f^{-L(i)}(x_i))^2$$

- Sea $L = \{1, 2, ..., n\} \to \{1, 2, ..., k\}$

-L(i) indice del subconjunto al que pertenece i.

- $f^L$ estimacion de f sin los datos del subconjunto L
- En LOO L(i) = i

### Validacion cruzada elegir k

- k=2: gran sesgo, solo usa la mitad de los datos.

-k = n, poco sesgo mucha varianza, las estimaciones son muy similares, mucho costo computacional.

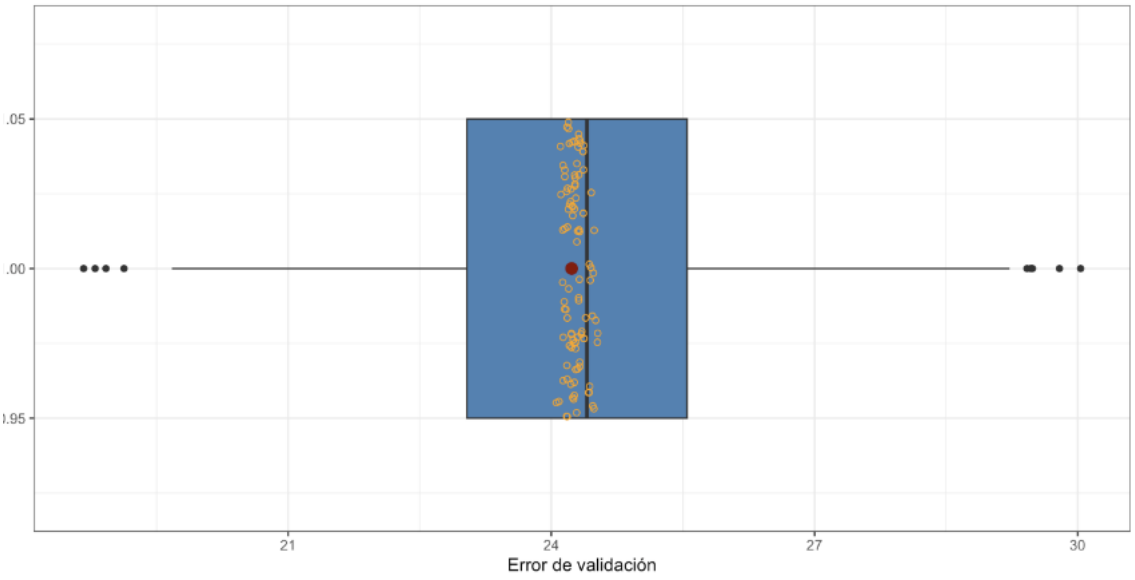
- k =5 o 10: menor costo computacional y compromiso entre sesgo y varianza, es lo que se utiliza.

Repetimos 100 veces el CV con k =10.

-Caja error de validacion, 1000 rep.

-Puntos naranjas, error CV (MSE), 100 rep con k =10.

- Punto rojo error CV LOO




## Parametros auxiliares

Habitualmente, modelos de ML tienen tunning parameters.

- Son "parametros" que deben ser provistos al algoritmo para que este pueda ser aplicado.

-Ejemplo: cuantos vecinos utilizamos? Este valor es necesario para aplicar el algoritmo

El modelo depende de parametros auxiliares $\hat{f}(x, \alpha)$

## Validacion cruzada:seleccion de parametros auxiliares

El modelo depende de parametros auxiliares: $\hat{f}(x, \alpha)$

$$CV(\hat{f}\alpha) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f^{l(i)}(x_i, \alpha))^2$$

-Obtenemos $CV(\hat{f}, \alpha)$ para valores de $\alpha$

-Usamos $\hat{\alpha}$ para varios valores de $\alpha$

- El modelo final es $f(x, \hat{\alpha})$ que se estima con todos los datos

## CV para obtener parametros auxiliares

- LLamamos parametro tunning o auxiliar, porque no queremos hacer inferencia sobre el. No hay que "estimarlo".
- Sin embargo, al usar CV, el valor $\hat{\alpha}$ depende de los datos.
- En ML, generalmente NO se toma en cuenta la variabilidad o incertidumbre sobre $\alpha$
- El supuesto implicito es que el algoritmo/modelo es muy flexible y pequenios cambios en $\alpha$ no afectan al ajuste.

## Cross - Validation en vecino mas cercano

Con los datos de publicidad: elegir cantidad **optima** de vecinos.

- Se elije el k* (optimo) con el MSE en cross validation mas chico.

## Regla 1se

Obtener el desvio del error CV para cada valor de k

Elegit el modelo mas simple con error por debajo de error(k*) + se(k)*

## Fijar parametros auxiliares

El metodo de vecino mas cercano (KNN) que utilizamos tiene un solo parametro auxiliar para fijar, que tiene un recorrido discreto y simple de recorrer.

- Muchos parametros(ejemplo enknn, funcion de distancia, medida de resumen)
- espacio mas dificil de explorar (ejemplo $\lambda$ en lasso)

**Existen distintas estrategias de busqueda**

-Grilla exahustiva

-Busqueda aleatoria

-Combinacion