

Aprendizaje Automático

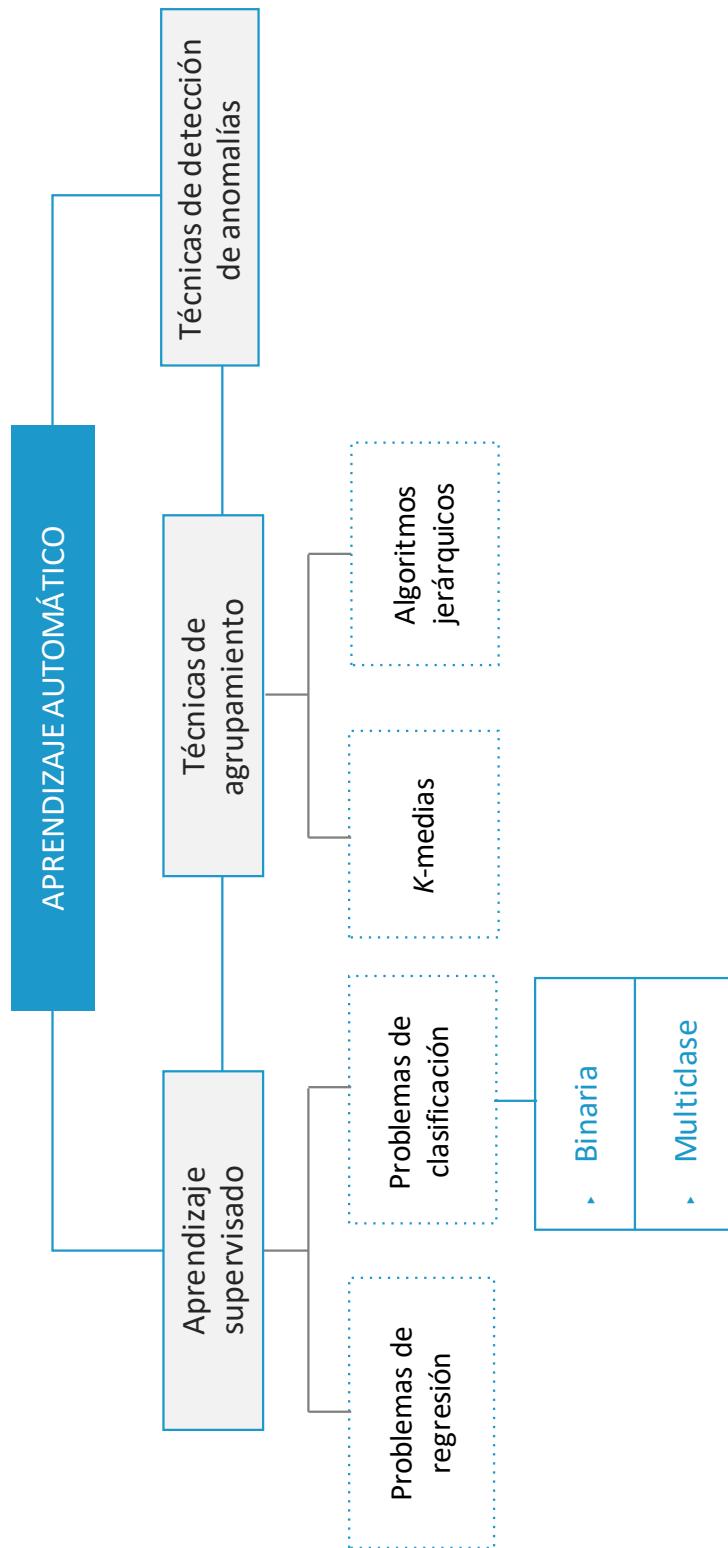
---

# Introducción al aprendizaje automático

# Índice

|   |           |
|---|-----------|
| <b>Esquema</b>  | <b>3</b>  |
| <b>Ideas clave</b>  | <b>4</b>  |
| 1.1. ¿Cómo estudiar este tema?                                | 4         |
| 1.2. Aprendizaje supervisado: problemas de regresión          |           |
|   | 5         |
| 1.3. Aprendizaje supervisado: problemas de<br>clasificación   | 6         |
| 1.4. Conjuntos de entrenamiento, test y validación<br>cruzada | 8         |
| 1.5. Técnicas de agrupamiento                                 | 11        |
| 1.6. Técnicas de detección de anomalías                       | 12        |
| 1.7. Referencias bibliográficas                               | 14        |
| <b>Lo + recomendado</b>                                       | <b>15</b> |
| <b>+ Información</b>  | <b>18</b> |
| <b>Test</b>   | <b>20</b> |

# Esquema



## 1.1. ¿Cómo estudiar este tema?

Estudia este tema a través de las **Ideas clave** disponibles a continuación.

**E**n este tema os introduciremos en los conceptos básicos del aprendizaje automático. Se entiende por **aprendizaje automático** aquellos algoritmos que se ejecutan en los ordenadores para aprender automáticamente en base a los datos proporcionados. Se trata de crear programas capaces de generalizar comportamientos a partir de los datos suministrados en forma de ejemplos. Es por tanto un proceso de inducción del conocimiento.

El aprendizaje automático, también conocido como *machine learning*, puede dividirse en algoritmos de **aprendizaje supervisado** y algoritmos de **aprendizaje no supervisado**. El aprendizaje supervisado utiliza ejemplos conocidos para obtener las inferencias mientras que el aprendizaje no supervisado no dispone de ejemplos con un objetivo o etiqueta conocido. A su vez, los **problemas** se pueden dividir en los siguientes cuatro sub-tipos:

- ▶ Aprendizaje supervisado: problemas de regresión.
- ▶ Aprendizaje supervisado: problemas de clasificación.
- ▶ Aprendizaje no supervisado: problemas de agrupamiento.
- ▶ Aprendizaje no supervisado: problemas de detección de anomalías.

En este tema se describen las **principales características y diferencias** de estos cuatro grupos de problemas que forman parte del aprendizaje automático. Además, se describen las diferencias entre los conjuntos de datos de entrenamiento, test y validación cruzada.

## 1.2. Aprendizaje supervisado: problemas de regresión

**E**l aprendizaje automático **surge a mediados de los años 80** con la aplicación de las redes de neuronas y los árboles de decisión. El aprendizaje automático se empezó a utilizar en problemas de predicción complejos donde los modelos estadísticos clásicos no eran muy buenos como, por ejemplo, el reconocimiento de voz e imágenes, la predicción de series temporales no lineales, la predicción de los mercados financieros, el reconocimiento de texto escrito, etc.

El **aprendizaje supervisado** es un tipo de aprendizaje automático donde se realizan inferencias por medio de una función que establece una correspondencia entre las entradas y la salida del sistema. En el aprendizaje supervisado tenemos datos que son generados por una «caja negra» donde un vector de variables de entrada  $x$  (llamadas variables independientes) entran por un lado y por otro lado las variables respuesta  $y$  son obtenidas.

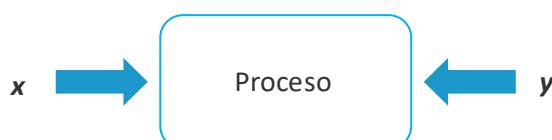


Figura 1. El aprendizaje automático busca el proceso que relaciona las variables de entrada  $x$  con las variables respuesta  $y$ .

En el aprendizaje supervisado para cada observación de las variables predictoras  $x$  existe una medida de la variable respuesta  $y$ . En el caso concreto de los problemas de regresión la variable respuesta  $y$  del sistema que se desea inferir o generalizar es una variable cuantitativa (numérica continua).

El objetivo del aprendizaje supervisado es **predecir las respuestas que habrá en el futuro** con nuevas variables de entrada. Para ello se utilizan algoritmos como modelos y se trata al mecanismo de generación de los datos como algo desconocido.

Es decir, se considera el interior de la caja como algo complejo y desconocido. Por tanto, el enfoque es buscar una función  $f(x)$  que opere con los datos  $x$  para producir las respuestas  $y$ .

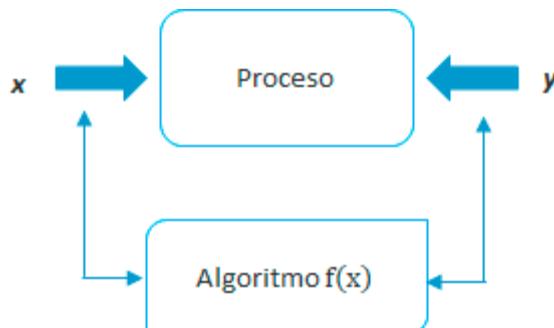


Figura 2. En el aprendizaje supervisado por medio de algoritmos se busca la función  $f(x)=y$  que relaciona la entrada con la salida.

La evaluación de este modelo se lleva a cabo por medio de la capacidad predictiva del modelo.

El **aprendizaje supervisado** tiene como objetivo generalizar las respuestas sobre datos no observados, utilizando para ello **ejemplos** observados previamente. En el caso de los problemas de **regresión** la variable respuesta  $y$  es una variable **numérica continua**.

### 1.3. Aprendizaje supervisado: problemas de clasificación

**E**l otro gran grupo de algoritmos de aprendizaje supervisado son los que están enfocados a problemas de clasificación. En los problemas de clasificación, el aprendizaje supervisado utiliza ejemplos conocidos para inferir la etiqueta (clasificar) de los vectores de entrada  $x$  eligiendo una de entre varias categorías o clases.

Estos algoritmos, al igual que en los problemas de regresión, utilizan ejemplos etiquetados previamente para «aprender» los patrones para llevar a cabo una clasificación.

En este tipo de problemas la variable respuesta *y* es una variable con dos o más categorías.

En los **problemas de clasificación**, utilizando aprendizaje supervisado, el objetivo es identificar a qué categoría pertenece una nueva observación utilizando para ello una serie de observaciones y categorías conocidas previamente.

Un ejemplo sería asignar a un correo electrónico la categoría de *spam* o no *spam* en función de los correos recibidos previamente. Otro ejemplo es realizar un diagnóstico a un paciente en función de sus características (sexo, presión sanguínea, colesterol, etc).

Los **problemas de clasificación** se dividen en dos grandes grupos: clasificación binaria y clasificación multi-clase. Los problemas de clasificación binaria buscan diferenciar las nuevas observaciones entre una de las dos clases posibles (ejemplo *spam* y no *spam*). Los problemas de clasificación multi-clase llevan asignar una nueva observación a una de entre varias clases posibles.

El **aprendizaje supervisado** tiene como objetivo generalizar utilizando para ello **ejemplos** conocidos. En el caso de los **problemas de clasificación** la variable respuesta *y* es **una variable con 2 o más categorías o clases**.

## 1.4. Conjuntos de entrenamiento, test y validación cruzada

La clave de cualquier modelo de aprendizaje automático es su capacidad de generalizar situaciones del futuro en función de los datos históricos observados. En la siguiente figura se describe el proceso de entrenamiento y predicción a alto nivel. En la fase de **entrenamiento** (a) se extraen las características o variables relevantes de los datos de entrada para construir un modelo por medio de algoritmos de aprendizaje automático. Posteriormente en la fase **predicción** (b) se realiza una extracción de variables similar sobre las que se aplica el modelo previamente entrenado para obtener el resultado estimado.

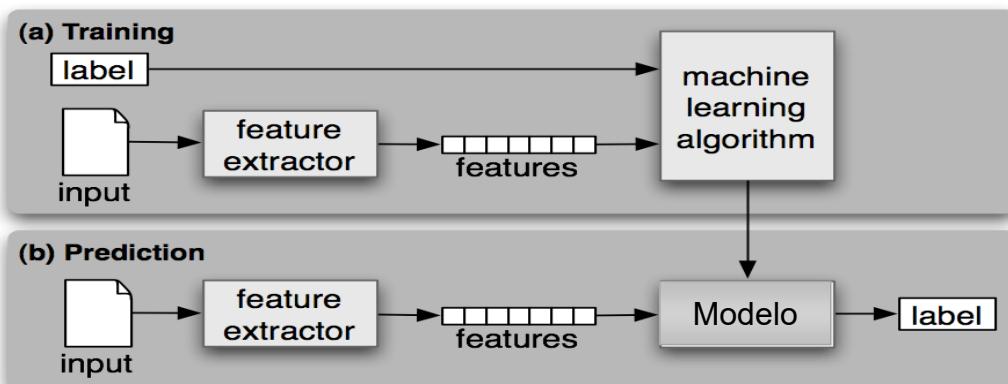


Figura 3. Ejemplo de las fases de entrenamiento (a) y predicción (b) en los procesos de aprendizaje automático. Recuperado de <http://www.nltk.org/book/ch06.html>

Durante la fase de construcción de los modelos de aprendizaje automático las diferentes implementaciones software proporcionan métricas de error. Estas métricas suelen obtenerse con el **conjunto de datos** utilizado para realizar el entrenamiento. Este conjunto de datos se conoce con el nombre de *training set* o **conjunto de entrenamiento**. Las métricas obtenidas con los datos de entrenamiento deben utilizarse solo como referencia, pues no son buenos indicadores del comportamiento futuro.

Un modelo puede ser capaz de tener un error mínimo con datos históricos (conjunto de entrenamiento) y no ser capaz de predecir bien los valores futuros. Por esta razón el error en el conjunto de entrenamiento suele ser un valor muy optimista y debe de ser interpretado con cautela.

Para solucionar el problema anterior, una buena práctica es utilizar un **conjunto de datos de test**. Este conjunto de datos de test puede estar formado con un subconjunto de los datos de entrenamiento.

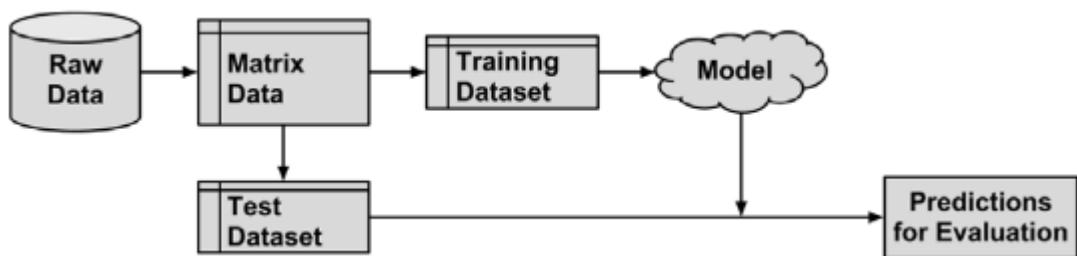


Figura 4. Ejemplo de utilización de un conjunto de test para evaluar el rendimiento de un modelo de aprendizaje supervisado.

El procedimiento de dividir los datos de los que se dispone entre conjuntos de entrenamiento y conjuntos de test, se conoce con el nombre de **hold-out**. En este procedimiento, el conjunto de entrenamiento se utiliza para crear el modelo que es evaluado utilizando el conjunto de test. Normalmente, se utiliza un 80 % de los datos para crear el conjunto de entrenamiento y un 20 % de los datos para generar conjunto de test. Para asegurarse que no existen diferencias sistemáticas entre los dos grupos es necesario obtener las instancias de forma aleatoria.

Es importante remarcar que para que este método sea riguroso, no se pueden utilizar observaciones o instancias del conjunto de test para crear el clasificador.

La repetición de la técnica de *hold-out* es la base para la técnica **de validación cruzada**. La técnica de validación cruzada es el estándar de la industria para estimar el rendimiento de los modelos. Esta técnica también es conocida como **k-fold**, donde

$k$  es un valor que indica que se han dividido los datos históricos en  $k$  particiones separadas llamadas *folds* o conjuntos.

Aunque  $k$  puede ser cualquier número, lo habitual es utilizar  $k=5$  o  $k=10$ . Esto es debido a que la evidencia empírica indica que hay poco beneficio en utilizar más de 10 *folds*. En este caso, para cada uno de los 10 *folds* (que comprenden un 10 % del total de los datos) se crea un modelo en los 9 *folds* restantes (90 % de los datos) y se evalúa en ese 10 %. Este proceso se realiza 10 veces y la media y la desviación típica de las ejecuciones es reportada.

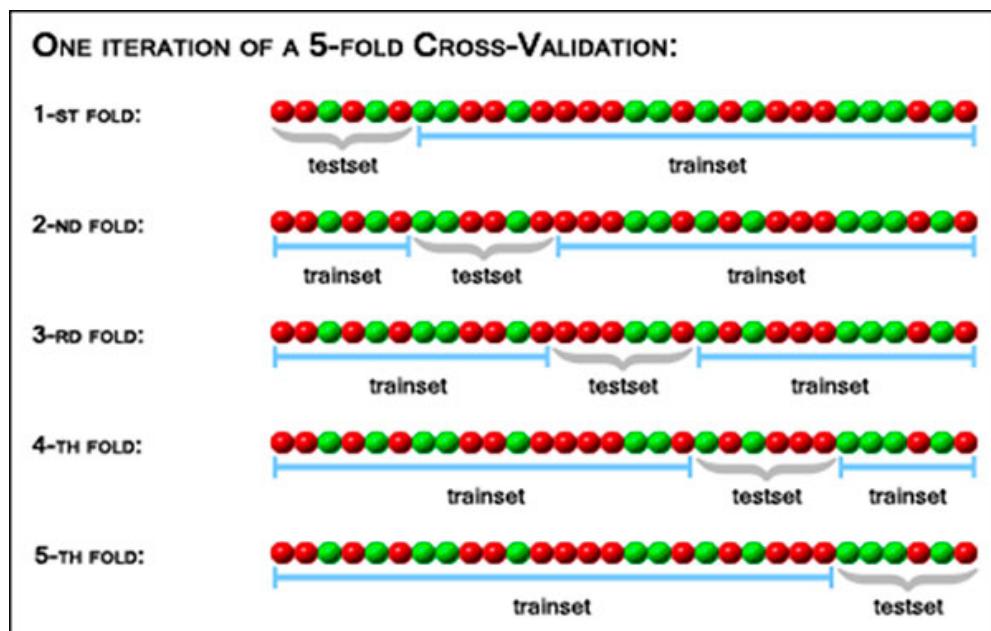


Figura 5. Ejemplo de división de un esquema de validación cruzada con 5 *folds*. Recuperado de <https://genome.tugraz.at/proclassify/help/pages/XV.html>

El conjunto de **entrenamiento** en aprendizaje supervisado proporciona un error que debe de ser evaluado con cautela. Es más riguroso evaluar a los modelos utilizando un conjunto separado e independiente llamado **conjunto de test**. Esta separación entre conjunto de test y entrenamiento se puede repetir a lo largo de los datos disponibles utilizando la técnica de **validación cruzada**.

## 1.5. Técnicas de agrupamiento

Las técnicas de agrupamiento o también conocidas como *clustering* son un ejemplo de **aprendizaje no supervisado** que se utilizan cuando desconocemos la etiqueta, clase o respuesta de las instancias de entrenamiento. En este tipo de problemas no se tiene conocimiento sobre las categorías o valores de los datos observados y se desea buscar la estructura oculta en los datos.

Por tanto, el proceso de modelado se lleva a cabo sobre un conjunto de ejemplos que **solo contiene las entradas del sistema**, y el algoritmo tiene que ser capaz de reconocer patrones para diferenciar entre los datos existentes. El resultado de estos algoritmos es una asignación de las observaciones a cada uno de los segmentos obtenidos.

La evaluación de este tipo de técnicas no es sencilla porque no se conoce el número de errores producidos. En concreto, cualquier método de evaluación sobre este tipo de técnicas no pueden calcularse con las etiquetas asignadas a las instancias sino con la separación que el algoritmo hace utilizando para ello métricas de similitud entre los miembros de cada una de las clases.

En la siguiente figura se muestra la separación entre clases que llevaría a cabo un algoritmo de agrupamiento para dos, tres y cuatro clases.

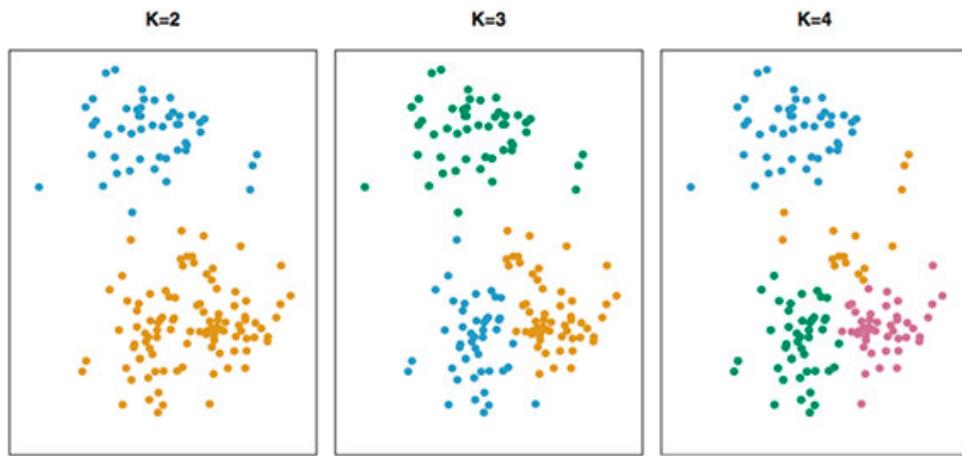


Figura 6. Ejemplos de agrupamiento para tres diferentes valores de k sobre los mismos datos. Fuente: James et al, 2013.

Las técnicas de agrupamiento son un ejemplo de **aprendizaje no supervisado** donde no se conocen las clases o valores de cada una de las instancias. El objetivo de estas técnicas es buscar la estructura oculta en los datos.

## 1.6. Técnicas de detección de anomalías

Una **anomalía** es una observación que es significativamente diferente del resto de observaciones. La **detección de anomalías** implica que esa observación es sospechosa de haber sido generada por un mecanismo diferente del resto de observaciones. La detección de anomalías es útil para dos objetivos diferentes. Por un lado, puede ser necesario eliminar estas anomalías de los datos para su posterior análisis. Por otro lado, el objetivo del análisis puede ser precisamente las anomalías y su origen.

Los algoritmos de detección de anomalías también son conocidos con el nombre de **detección de outliers**. Estos problemas tienen gran aplicación en entornos como el fraude, la detección de intrusos, la detección de defectos estructurales, problemas médicos o errores en texto.

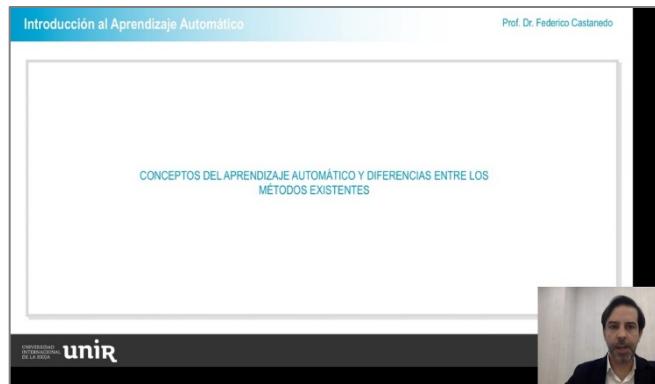
Los algoritmos de detección de anomalías se pueden clasificar entre:

- ▶ Detección de anomalías supervisada.
- ▶ Detección de anomalías no supervisada.

La detección de anomalías de forma no supervisada busca anomalías en conjuntos de datos sin etiquetar con la hipótesis de que la mayoría de las instancias en el conjunto de datos son normales y observando aquellas que menos se parecen a la mayoría. Por otro lado, la detección de anomalías supervisada utiliza instancias o ejemplos previamente etiquetados como «normales» y «anormales» para entrenar un clasificador.

La detección de anomalías **busca observaciones que son significativamente diferentes** del resto de las observaciones. Esta detección puede hacerse utilizando **aprendizaje no supervisado**, si no conocemos ejemplos previos de anomalías. O bien utilizando **aprendizaje supervisado** para entrenar un clasificador que distinga entre ejemplos normales o anormales.

En el siguiente vídeo afianzaremos los conceptos introductorios del aprendizaje automático y las principales diferencias entre los métodos existentes.



Conceptos del aprendizaje automático y diferencias entre los métodos existentes.

---

Accede al vídeo a través del aula virtual

---

## 1.7. Referencias bibliográficas

James G., Witten, D., Hastie, T and Tibshirani, R. (2013). *An Introduction to Statistical Learning with Applications in R*. Springer.

# Lo + recomendado

## No dejes de leer

### A Few Useful Things to Know about Machine Learning

Domingos, P. (2012). A Few Useful Things to Know About Machine Learning. University of Washington.

Artículo de Pedro Domingos que explica conceptos importantes sobre *machine learning*.

---

Accede al artículo a través del aula virtual en la siguiente dirección web:

<https://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf>

---

# No dejes de ver

## Train/Test en Scikit-learn

Vídeo que describe el proceso de generar conjuntos de *train* y *test* con Python.

**3.1. Cross-validation: evaluating estimator performance**

Learning the parameters of a prediction function and testing it on the same data is a methodological mistake: a model that just repeat the labels of the samples that it has just seen would have a perfect score but would fail to predict anything useful on yet-unseen data. This situation is called **overfitting**. To avoid it, it is common practice when performing a (supervised) machine learning experiment to hold out part of the available data as a **test set** `X_test, y_test`. Note that the word "experiment" is not intended to denote academic use only, because even in commercial settings machine learning usually starts out experimentally.

In scikit-learn a random split into training and test sets can be quickly computed with the `train_test_split` helper function. Let's load the iris data set to fit a linear support vector machine on it:

```
>>> import numpy as np
```

---

Accede al vídeo a través del aula virtual o desde la siguiente dirección web:

<https://www.youtube.com/watch?v=lSwvUmZCvco>

---

## Validación cruzada en R

Vídeo corto que describe el proceso de validación cruzada en *R*.



---

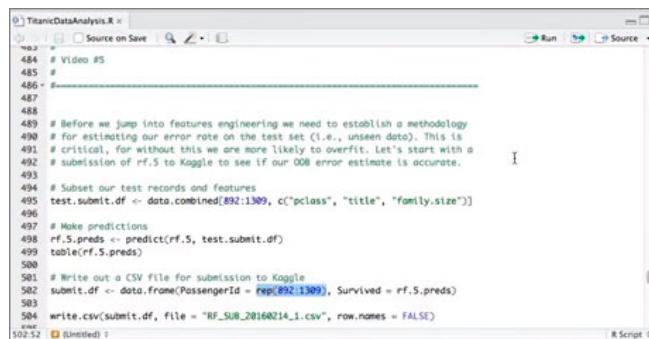
Accede al vídeo a través del aula virtual o desde la siguiente dirección web:

<https://www.youtube.com/watch?v=OwPQHmjJURI>

---

## Validación cruzada en R (vídeo largo)

Vídeo largo que describe el proceso de validación cruzada utilizando el lenguaje *R*.



```
482 # Video #5
483 #
484 # Before we jump into features engineering we need to establish a methodology
485 # for estimating our error rate on the test set (i.e., unseen data). This is
486 # critical, for without this we are more likely to overfit. Let's start with a
487 # submission of rf.5 to Kaggle to see if our OOB error estimate is accurate.
488
489 # Subset our test records and features
490 test.submit.df <- data.combined[892:1309, c("pclass", "title", "family.size")]
491
492 # Make predictions
493 rf.5.preds <- predict(rf.5, test.submit.df)
494
495 # Write out a CSV file for submission to Kaggle
496 submit.df <- data.frame(PassengerId = rep(892:1309), Survived = rf.5.preds)
497
498 write.csv(submit.df, file = "RF_SUB_20160214_1.csv", row.names = FALSE)
499
500
```

---

Accede al vídeo a través del aula virtual o desde la siguiente dirección web:

<https://www.youtube.com/watch?v=84JSk36og34>

---

## Train/Dev/Test Set Distribution

Vídeo en detalle de Andrew Ng explicando los conceptos de conjunto de entrenamiento y conjunto de test.



---

Accede al vídeo a través del aula virtual o desde la siguiente dirección web:

<https://www.youtube.com/watch?v=M3qplzy4MQk>

---

## Webgrafía

### Jupyter

Página web sobre bias, variance and cross-validation en Python.



---

Accede a la página web a través del aula virtual o desde la siguiente dirección web:

<http://nbviewer.jupyter.org/github/cs109/content/blob/master/labs/lab5/Lab5.ipynb>

---

## Bibliografía

Breiman, L. (2001). Statistical Modeling: The two Cultures. *Statistical Science*. 16(3), 199-231. Recuperado de <https://projecteuclid.org/euclid.ss/1009213726>

Bzdok, D., Altman, N. y Krzywinski, M. (2018). Points of Significance: Statistics versus machine learning. *Nature Methods*, 233-234.

Domingos, P. (2012). A Few Useful Things to Know About Machine Learning.  
University of Washington. Recuperado de  
<https://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf>

Norvig, P. Y Russel, S. (2004). *Inteligencia Artificial: Un enfoque moderno*. Madrid:  
Pearson, Prentice/Hall.

- 1.** ¿Cuáles de las siguientes afirmaciones son correctas?

  - A. El aprendizaje automático utiliza siempre ejemplos con clases conocidas previamente.
  - B. El aprendizaje automático sirve únicamente para resolver problemas de predicción numérica.
  - C. El aprendizaje supervisado busca automáticamente los mecanismos que relacionan una entrada con una salida.
  - D. B y C son correctas.
- 2.** En el caso de los progresos de regresión:

  - A. La variable respuesta que se desea predecir es de tipo cualitativa.
  - B. La variable respuesta que se desea predecir es de tipo cuantitativa.
  - C. No siempre existe una variable respuesta.
  - D. Ninguna de las anteriores es correcta.
- 3.** En los problemas de clasificación:

  - A. La variable respuesta contiene siempre más de dos categorías.
  - B. La variable respuesta contiene siempre dos o más categorías.
  - C. La variable respuesta es de tipo numérico.
  - D. Ninguna de las anteriores es correcta.
- 4.** En la fase de entrenamiento de los modelos:

  - A. Se realiza la extracción de características y se utiliza para generar posteriormente una predicción.
  - B. Se elige qué modelo es el mejor.
  - C. Se aprende un modelo que podrá ser utilizado posteriormente.
  - D. Ninguna de las anteriores es correcta.

**5.** En el aprendizaje automático

- A. El conjunto de entrenamiento se utiliza para construir un modelo.
- B. El conjunto de test se utiliza para evaluar un modelo.
- C. Si un modelo tiene un error mínimo en el conjunto de entrenamiento también lo tendrá en el conjunto de test.
- D. Todas las anteriores son correctas.

**6.** A la hora de construir los conjuntos de entrenamiento y test:

- A. Es necesario que no haya diferencias sistemáticas entre uno y otro.
- B. Es necesario que haya diferencias sistemáticas entre uno y otro.
- C. Ninguna de las anteriores es correcta.

**7.** El método de *hold-out*:

- A. Consiste en separar los datos en  $k$  particiones distintas.
- B. Consiste en separar los datos disponibles en entrenamiento y test.
- C. Ninguno de los anteriores es correcto.

**8.** El método k-cross validation:

- A. Consiste en dividir los datos disponibles  $k$  grupos de tamaño variable cada uno de ellos.
- B. Consiste en dividir los datos disponibles en  $k$  grupos del mismo tamaño.
- C. Ninguna de las anteriores es correcta.

**9.** Las técnicas de agrupamiento:

- A. Se utilizan para agrupar los datos cuando se conoce el valor de las clases.
- B. Se utilizan para agrupar los datos cuando no se conoce el valor de las clases.
- C. Se utiliza para clasificar las variables en función de una clase target.
- D. Ninguna de las anteriores es correcta.

**10.** Los algoritmos de detección de anomalías:

- A. Siempre son algoritmos de detección supervisada.
- B. Siempre son algoritmos de detección no supervisada.
- C. Se puede clasificar entre algoritmos de detección supervisada y no supervisada.

Aprendizaje Automático

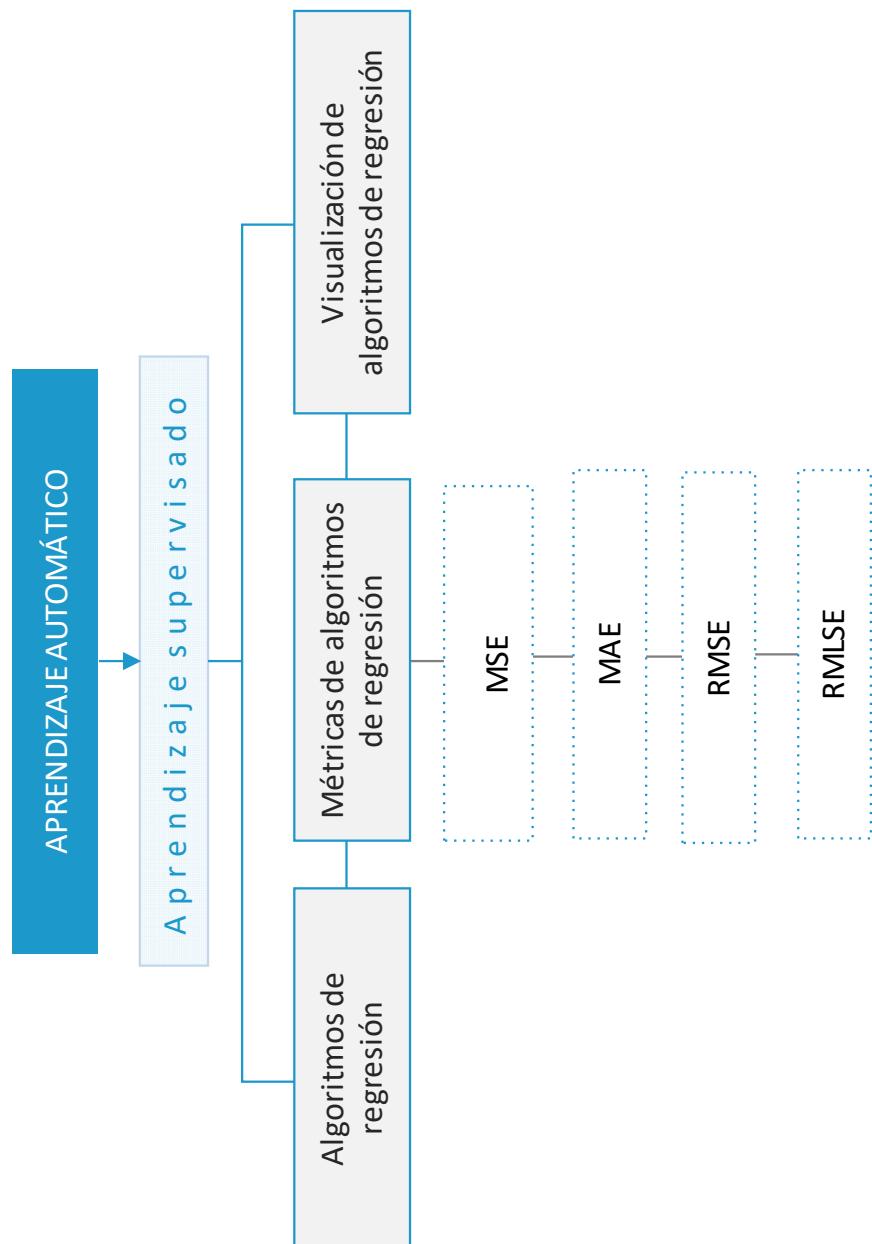
---

# Evaluación de algoritmos de regresión

# Índice

|                                   |           |
|-----------------------------------|-----------|
| <b>Esquema</b>                    | <b>3</b>  |
| <b>Ideas clave</b>                | <b>4</b>  |
| 2.1. ¿Cómo estudiar este tema?    | 4         |
| 2.2. Algoritmos de regresión      | 4         |
| 2.3. Métricas de error            | 10        |
| 2.4. Visualización de los errores | 12        |
| <b>Lo + recomendado</b>           | <b>14</b> |
| <b>+ Información</b>              | <b>17</b> |
| <b>Test</b>                       | <b>18</b> |

# Esquema



## 2.1. ¿Cómo estudiar este tema?

Estudia este tema a través de las **Ideas clave** disponibles a continuación.

**E**ste tema profundiza en los algoritmos de aprendizaje supervisado, haciendo el foco en los **algoritmos de regresión** los cuales realizan una estimación o predicción de una variable numérica o cuantitativa.

Como ya se ha definido, el problema de regresión es aquel en el cual la variable a predecir y es una variable numérica o cuantitativa.

El objetivo de este tema es obtener una mayor comprensión sobre los algoritmos de regresión, utilizar las métricas de error más comunes y ser capaz de visualizar los errores de forma gráfica.

- ▶ En primer lugar, se describe el funcionamiento general de los algoritmos de regresión.
- ▶ A continuación, se describen las métricas comunes utilizadas por este tipo de algoritmos.
- ▶ Finalmente se describen visualización que permite obtener de forma gráfica una evaluación del funcionamiento de estos algoritmos.

## 2.2. Algoritmos de regresión

Dentro del campo del aprendizaje automático, los **algoritmos de regresión** son un tipo concreto de **algoritmos de aprendizaje supervisado** y consisten en realizar una

predicción de una variable numérica o cuantitativa. En el aprendizaje supervisado, para cada una de las observaciones de las variables predictoras ( $x_i$ ) existe una medida de la variable respuesta ( $y_i$ ). Se desea crear un **modelo que relacione las variables y las respuestas con el objetivo de predecir las respuestas de observaciones en el futuro.**

Existen numerosos problemas del mundo real donde la variable que se desea predecir o estimar es una variable numérica, por ejemplo, los ingresos de una persona, el precio de venta de un inmueble o algo tan dispar como la fuerza del cemento. De forma general, el **aprendizaje supervisado se puede utilizar** en todos aquellos **casos en los cuales existe conocimiento de un valor cuantitativo previo.**

La **terminología o notación más común** es la siguiente:

- ▶ **Variable respuesta:** también conocida como *outcome*, variable **dependiente**, variable **objetivo**, *target*, *class*, etc. Se suele denotar por  $y$ .
- ▶ Vector de  $p$  **mediciones predictoras** llamado  $x$ : también conocido como *inputs*, *regressors*, *features*, **variables**, **variables independientes**, etc.
- ▶ Se tienen datos de entrenamiento conocidos como **training data**:  $(x_1, y_1), (x_N, y_N)$  que son observaciones, ejemplos o instancias de cada una de las medidas de entrada.

Para entender el funcionamiento de los algoritmos de regresión, observemos el siguiente gráfico:

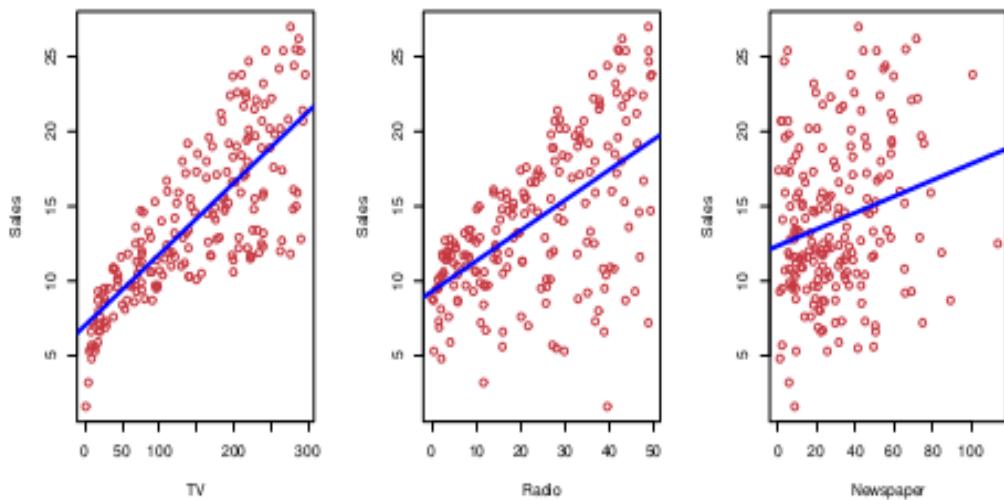


Figura 1. El gráfico muestra las ventas en unidades de miles en función del gasto en TV, radio y periódico en miles de dólares para 200 mercados diferentes. La línea azul representa un modelo simple que se puede utilizar para predecir las ventas utilizando cada una de estas variables más importantes. Fuente: James et al., 2013.

El gráfico anterior muestra la distribución de las ventas en función del gasto publicitario en TV, radio o prensa escrita. Es posible determinar cada uno de los valores del eje y en función del eje x utilizando cada una de las gráficas de forma independiente, o bien combinar las tres para obtener una función de las ventas obtenidas en función de las tres entradas.

$$Sales \approx f(TV, Radio, Newspaper)$$

Lo cual se puede definir utilizando notación vectorial de la siguiente forma:

$$X = \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} \quad Y = f(x) + \varepsilon$$

Supongamos que estamos en un plano de dos dimensiones. Se define como función  $f(x)$  ideal a la curva que mejor aproxima los puntos.

Supongamos que tenemos una situación como la de la gráfica de abajo:

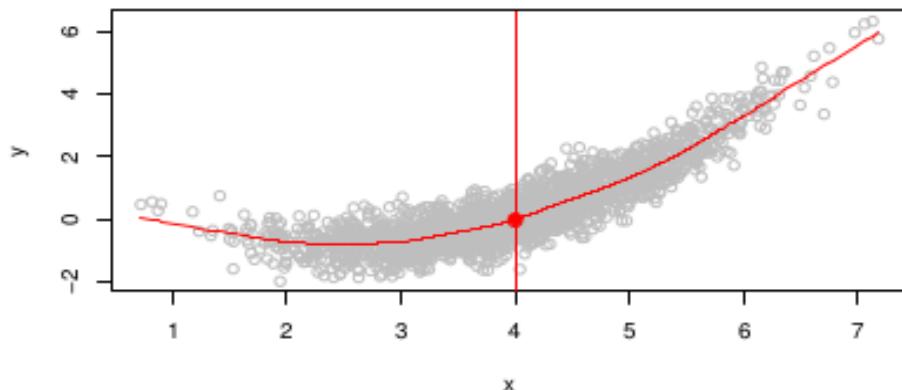


Figura 2. Función de regresión pintada en color rojo sobre la nube de puntos, donde se observa que existen diversos valores de  $y$  para un mismo valor del eje  $x$ .

Recuperado de: <https://lagunita.stanford.edu/courses/HumanitiesSciences/StatLearning/Winter2016/about>

Ahora, supongamos que queremos obtener el valor de  $y$  en el punto  $x=4$ , ¿Qué sucede? Pues que tenemos varios puntos con valores de  $y$  diferentes para el punto  $x=4$ . Este valor en el punto  $x=4$  se define de la siguiente forma:

$$f(4) = E(Y|X = 4) \quad \text{Probabilidad de Y sabiendo que } x = 4$$

A la función  $f(x)$  ideal también se la conoce con el nombre de función de regresión, la cual para todos los puntos del eje  $x$  se define de la siguiente forma:

$$F(x) = E(Y|X = x)$$

Esta función también se encuentra definida para un vector de  $x$ :

$$f(x) = f(x_1, x_2, x_3) = E(Y|X_1 = x_1, X_2 = x_2, X_3 = x_3)$$

La función predictora óptima de  $y$  es aquella con un error cuadrático medio (MSE) menor:

$$f(x) = E(Y|X = x)$$

Por tanto, que minimiza:

$$E[(Y - g(X))^2 | X = x]$$

Para todas las funciones  $g$  en todos los puntos  $X=x$

Este cálculo se divide en dos términos distintos conocidos como error reducible y error irreducible:

$$E[(Y - g(X))^2 | X = x]E[Y - \hat{f}(X)]^2 | X = x] = [f(x) - \hat{f}(x)]^2 + Var(\epsilon)$$

¿Cómo estimar el  $f(x)$  ideal?

El problema es que normalmente tendremos pocos o ningún punto en  $x=4$ , por tanto no se puede calcular directamente  $E(Y|X = x)$  y la solución es relajar la definición de  $f(x)$  ideal, siendo por tanto necesario calcular:  $\hat{f}(x) = Ave(Y|X \in N(x))$ , donde  $N(x)$  es un vecindario de puntos cercanos de  $x$ .

Un ejemplo ilustrativo de esta situación se puede observar en la siguiente gráfica:

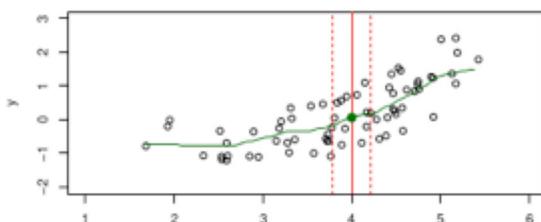


Figura 3. Estimación del valor de un punto en función del valor de sus vecinos (denotados por las líneas punteadas). Recuperado de

<https://lagunita.stanford.edu/courses/HumanitiesSciences/StatLearning/Winter2016/about>

Esto es lo que se conoce como estimación *nearest-neighbor average*. Por lo general, el problema suele ser que estimar los puntos cercanos utilizando *nearest-neighbor average* funciona bien para problemas con pocas dimensiones ( $p$  pequeña) y muchos

ejemplos ( $N$  muy grande), pues de lo contrario se produce el problema de la *curse of dimensionality*.

El problema de la *curse of dimensionality* implica que puntos cercanos en dimensiones pequeñas se van alejando de forma exponencial a medida que se incrementa el número de dimensiones.

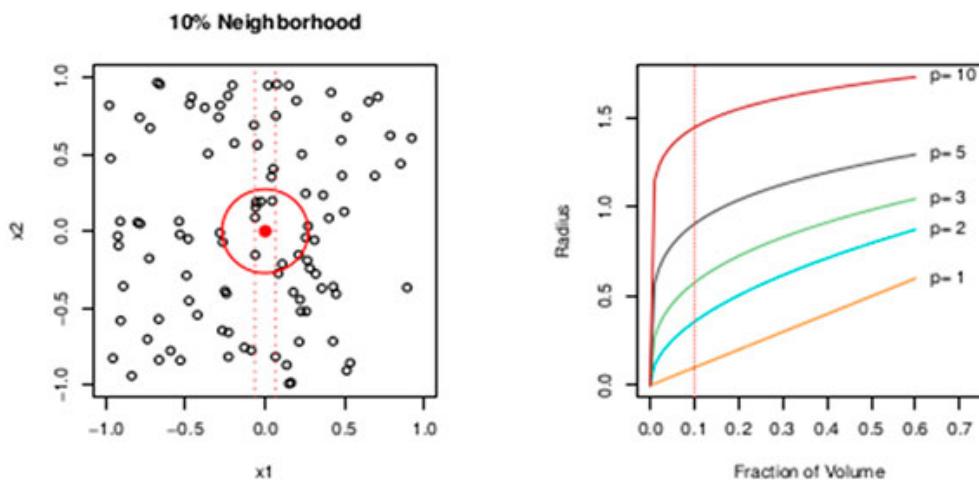


Figura 4. Efecto del problema de *curse of dimensionality*. Recuperado de <https://lagunita.stanford.edu/courses/HumanitiesSciences/StatLearning/Winter2016/about>

Este efecto se observa en la figura anterior: en el gráfico de la izquierda en una dimensión es posible cubrir el 10 % de los puntos con las líneas punteadas del eje  $x_1$ . En el caso de dos dimensiones, el 10 % de los puntos se cubren con el área interior del círculo rojo. En el gráfico de la derecha se observa este efecto para los casos hasta diez dimensiones ( $p=10$ ). El efecto consiste en que cada vez que se incrementa una dimensión para tener  $p$  dimensiones, los puntos que estaban cercanos en  $p-1$  dimensiones pasan a estar más alejados.

En los algoritmos de regresión se utiliza una variable respuesta, un vector de  $p$  mediciones y datos de entrenamiento para construir una función de regresión. La función de regresión consta de un error reducible y un error irreducible. La estimación de la función  $f(x)$  ideal puede conllevar los problemas de *curse of dimensionality*.

## 2.3. Métricas de error

**E**xisten diferentes **métricas de error** que se pueden aplicar en un **problema de regresión** para obtener la función  $f(x)$  ideal. Las más comunes o habituales y sus definiciones matemáticas son:

- ▶ **Error cuadrático medio, mean square error (MSE):** se define como la media de la diferencia entre el valor real y el valor predicho o estimado al cuadrado.

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

- ▶ **Error absoluto medio, mean absolute error (MAE):** se define como la diferencia en valor absoluto entre el valor real y el valor predicho.

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

- ▶ **Raíz del error cuadrático medio, root mean square Error (RMSE):** se define como la raíz cuadrada de la media de la diferencia entre el valor real y el valor predicho o estimado al cuadrado.

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2} = \text{root}(MSE)$$

Esta métrica comparada con el error absoluto medio (MAE) amplifica y penaliza los errores grandes. Por otro lado, en MAE cada error contribuye al total del error en función de su valor absoluto. El siguiente código muestra un ejemplo de cálculo en R:

```
RMSE <- sqrt(mean((y-y_pred)^2))
```

Por otro lado, el siguiente código muestra un ejemplo de cálculo en Python:

```
from sklearn.metrics import  
    mean_squared_error  
RMSE = mean_squared_error(y,  
                           y_pred)**0.5
```

- Logaritmo de la raíz del error cuadrático medio, *root mean logarithmic square error (RMLSE)*:

$$RMLSE = \sqrt{1/n \sum_{i=1}^n (\log (y_i + 1) - \log (\hat{y}_i + 1))^2}$$

Esta métrica penaliza una *under-prediction* más que una *over-prediction*. El siguiente código de Python muestra un ejemplo de cómo calcular esta métrica.

```
import math  
  
def rmsle(y, y_pred):  
    assert len(y) == len(y_pred)  
    terms_to_sum = [(math.log(y_pred[i] + 1) -  
                    math.log(y[i] + 1)) ** 2.0 for i, pred in  
                    enumerate(y_pred)]  
    return (sum(terms_to_sum) * (1.0 / len(y))) ** 0.5
```

Las métricas más comunes de los problemas de regresión, son el error cuadrático Medio (**MSE**), error absoluto medio (**MAE**), raíz del error cuadrático medio (**RMSE**) y logaritmo de la raíz del error cuadrático medio (**RMLSE**).

## 2.4. Visualización de los errores

Una de las mejores formas de evaluar un modelo de regresión es utilizando gráficos. Uno de los gráficos que más información proporcionan visualmente es un **diagrama de dispersión de dos dimensiones** donde el eje x son los valores reales y el eje y los valores predichos o estimados (o viceversa). En el siguiente gráfico se muestra un ejemplo.

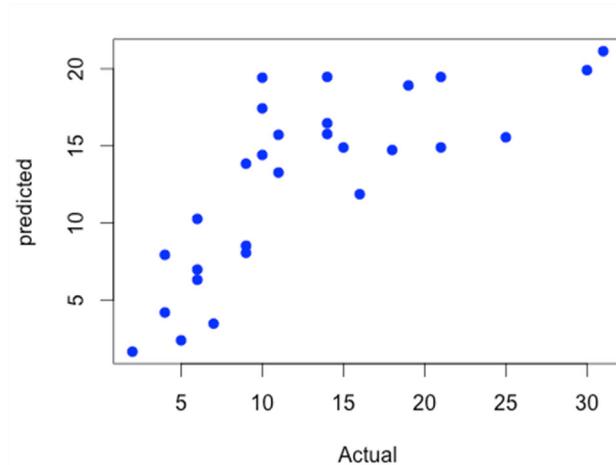


Figura 5. Gráfico de dispersión que muestra la relación y el error entre las predicciones y los valores reales.  
Fuente: <https://stats.stackexchange.com/questions/104622/what-does-an-actual-vs-fitted-graph-tell-us>

Cada uno de los puntos corresponde a una de las estimaciones o predicciones realizadas. En el caso de que las **predicciones o estimaciones fueran perfectas**, caerían directamente sobre la diagonal del gráfico. Por otro lado, los puntos que caen por encima de la diagonal son **sobre estimaciones de las predicciones**, mientras que los puntos que caen por debajo de la diagonal son **predicciones que se quedan cortas**.

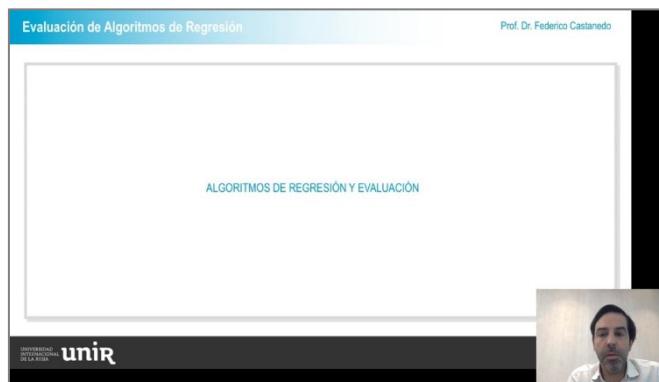
Una métrica común que se utiliza para medir la dispersión en este tipo de diagramas es el **coeficiente de correlación**, el cual cuantifica la relación lineal entre dos variables. Este coeficiente toma valores entre -1 y 1, donde 1 implica correlación lineal positiva de forma completa, -1 correlación lineal negativa de forma completa; y cuanto más cercano de 0 sea el valor menor correlación entre las dos variables.

Una correlación lineal positiva implica que cuando el valor en el eje x crece, el valor en el eje y lo hace en la misma proporción. Por el contrario, una correlación lineal negativa implica que cuando el valor en el eje x crece el valor en y decrece de forma proporcional. Esta relación se define matemáticamente con la siguiente función.

Esta bondad de ajuste también se puede medir utilizando el **coeficiente de determinación  $r^2$** . Este coeficiente indica la fracción de la variación explicada por la recta de regresión respecto a la variación total. Coincide con el cuadrado del coeficiente de correlación y puede tomar valores entre 0 y 1, siendo 1 el mejor ajuste y 0 el peor. De forma matemática se define como:

Una de las mejores formas de visualización del resultado de un modelo de regresión es el obtenido por medio de un *scatter plot*, donde en el eje x aparezca la predicción y en el eje y el valor real.

A continuación, en este vídeo se enfatizarán los primeros conceptos de regresión y cómo se pueden evaluar.



Algoritmos de regresión y evaluación.

---

Accede al vídeo a través del aula virtual

---

# Lo + recomendado

## No dejes de leer

### Difference Between Classification and Regression in Machine Learning

Brownlee, J. (diciembre, 2017). Difference Between Classification and Regression in Machine Learning.

Explicación sobre las diferencias entre clasificación y regresión en aprendizaje automático.

Accede al artículo a través del aula virtual o desde la siguiente dirección web:

<https://machinelearningmastery.com/classification-versus-regression-in-machine-learning/>

### Classification Versus Regression—Intro to Machine Learning#5

Fumo, D. (13 de febrero de 2017). Classification Versus Regression—Intro to Machine Learning#5.

Explicación sobre los métodos de clasificación *versus* regresión

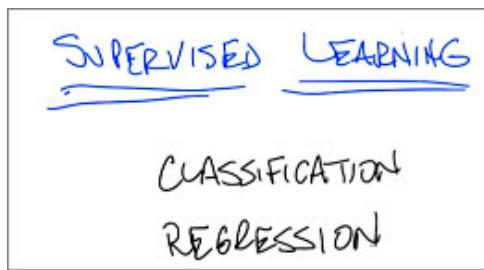
Accede al artículo a través del aula virtual o desde la siguiente dirección web:

<https://medium.com/simple-ai/classification-versus-regression-intro-to-machine-learning-5-5566efd4cb83>

## No dejes de ver

### Difference between Classification and Regression Georgia Tech-Machine Learning

Vídeo que comenta las diferencias entre los problemas de clasificación y regresión.



---

Accede al vídeo a través del aula virtual o desde la siguiente dirección web:

<https://www.youtube.com/watch?v=i04Pfrb71vk>

---

### Analyzing and modeling complex and big data

Vídeo de TEDx que describe los retos de analizar datos en entornos Big Data.



---

Accede al vídeo a través del aula virtual o desde la siguiente dirección web:

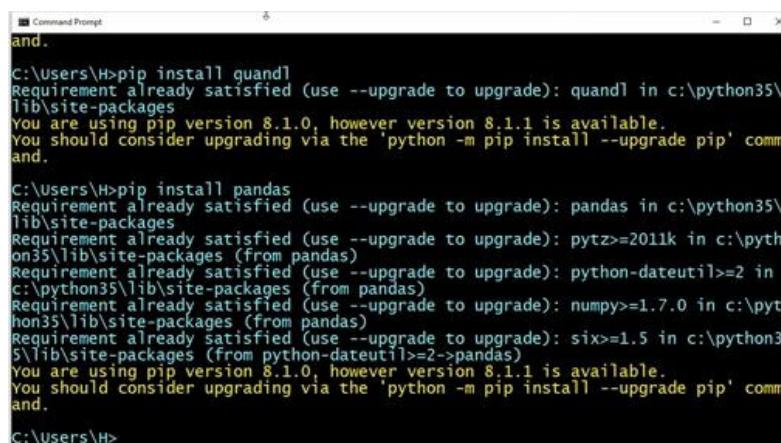
<https://www.youtube.com/watch?v=8DqQCZMawNg>

---

## Introducción Regresiones – Tutorial Aprendizaje de Máquina práctico con Python

p.2

Tutorial de regresión lineal con *scikit-learn* y Python



```
C:\Users\H>pip install quandl
Requirement already satisfied (use --upgrade to upgrade): quandl in c:\python35\lib\site-packages
Requirement already satisfied (use --upgrade to upgrade): pandas in c:\python35\lib\site-packages
Requirement already satisfied (use --upgrade to upgrade): pytz>=2011k in c:\python35\lib\site-packages (from pandas)
Requirement already satisfied (use --upgrade to upgrade): python-dateutil>=2 in c:\python35\lib\site-packages (from pandas)
Requirement already satisfied (use --upgrade to upgrade): numpy>=1.7.0 in c:\python35\lib\site-packages (from pandas)
Requirement already satisfied (use --upgrade to upgrade): six>=1.5 in c:\python35\lib\site-packages (from python-dateutil>=2->pandas)
Requirement already satisfied (use --upgrade to upgrade): You are using pip version 8.1.0, however version 8.1.1 is available.
Requirement already satisfied (use --upgrade to upgrade): You should consider upgrading via the 'python -m pip install --upgrade pip' command.

C:\Users\H>
```

---

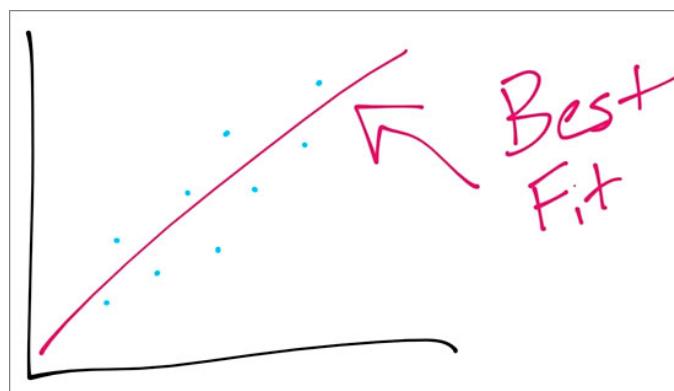
Accede al vídeo a través del aula virtual o desde la siguiente dirección web:

<https://www.youtube.com/watch?v=JcI5Vnw0b2c>

---

## Regression How it Works – Practical Machine Learning Tutorial with Python p.7

Tutorial de regresión lineal con Python



---

Accede al vídeo a través del aula virtual o desde la siguiente dirección web:

<https://www.youtube.com/watch?v=V59bYfIomVk>

---

## Bibliografía

Hastie, T., Tibshirani R., Friedman, J. (2009). *The Elements of Statistical Learning*. Second edition. Nueva York: Springer.

James G., Witten, D., Hastie, T and Tibshirani, R. (2013). *An Introduction to Statistical Learning with Applications in R*. Nueva York: Springer.

Pino, A. E., Chichande, B. S. y Tovar, Y. J. (2019). Determinación de modelos predictivos para los indicadores de competitividad empresarial aplicando regresión lineal. *Revista Ibérica de Sistemas e Tecnologías de Información*, 94-107.

1. En el aprendizaje supervisado:

- A. Para cada una de las observaciones puede existir o no una medida de la variable respuesta.
- B. Para cada una de las observaciones existe una medida de la variable respuesta.

2. Con cuáles de los siguientes nombres se conoce a la variable respuesta:

— A. Estimador.

— B. Variable objetivo.

— C. *Target*.

— D. *Regressors*.

3. Con cuáles de los siguientes nombres se conocen a las variables predictoras:

— A. *Inputs*.

— B. Variables independientes.

— C. Modelos.

— D. *Features*.

4. A la función  $f(x)$  ideal se conoce con el nombre de:

? — A. Función de regresión.

— B. Función estimadora.

— C. Función perfecta.

5. La función predictora óptima es aquella:
- A. Que se obtiene más rápidamente.
  - B. Tiene un error cuadrático medio menor.
  - C. Abarca el mayor número de ejemplos.
6. El cálculo del error en la función predictora óptima se divide en:
- A. Error reducible y error irreducible.
  - B. El error aleatorio y el error futuro.
  - C. Ninguna de las anteriores es correcta.
7. Cuáles de las siguientes métricas se utilizan para evaluar los problemas de regresión:
- A. MAE.
  - B. MSE.
  - C. RMSE.
  - D. RMSLE.
8. El *root mean square error* (RMSE):
- A. Penaliza los errores pequeños.
  - B. Amplifica y penaliza los errores grandes.
  - C. Los errores grandes y pequeños se tratan igual.
9. El *root mean square logarithmic error* (RMSLE):
- A. Penaliza más un *under-prediction* que un *over-prediction*.
  - B. Penaliza más un *over-prediction* que un *under-prediction*.
  - C. Ninguna de las anteriores es correcta.
10. El coeficiente de determinación permite:
- A. Obtener una métrica de la bondad del ajuste.
  - B. Determinar la varianza en la predicción de los datos.
  - C. Todas las anteriores son correctas.

Aprendizaje Automático

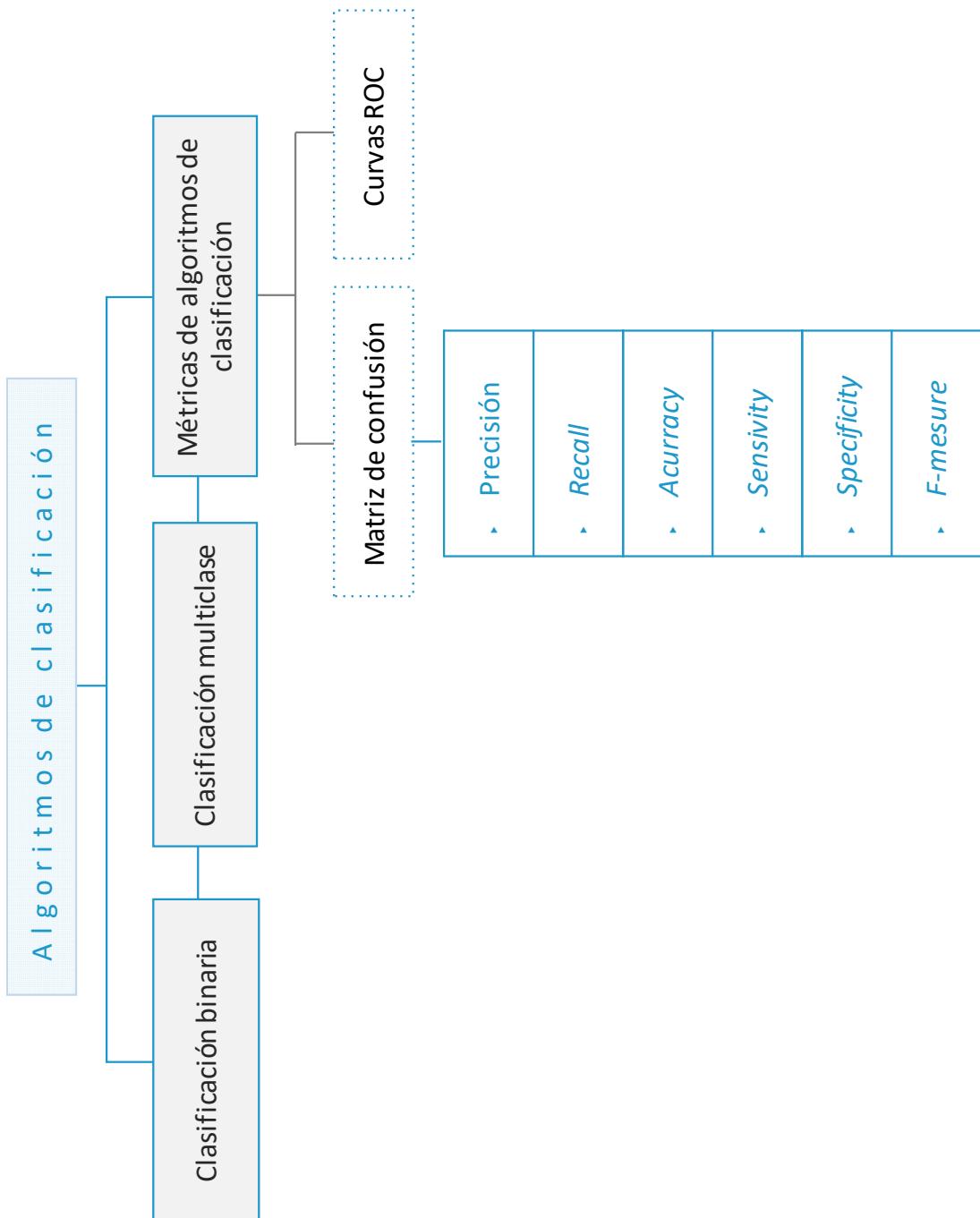
---

# Evaluación de algoritmos de clasificación

# Índice

|  |           |
|--|-----------|
| <b>Esquema</b>                                   | <b>3</b>  |
| <b>Ideas clave</b>                               | <b>4</b>  |
| 3.1. ¿Cómo estudiar este tema?                   | 4         |
| 3.2. Algoritmos de clasificación                 | 5         |
| 3.3. Métricas de evaluación: matriz de confusión | 5         |
| 3.4. Métricas de evaluación: curvas ROC, AUC     | 13        |
| <b>Lo + recomendado</b>                          | <b>17</b> |
| <b>+ Información</b>                             | <b>19</b> |
| <b>Test</b>                                      | <b>21</b> |

# Esquema



## 3.1. ¿Cómo estudiar este tema?

Estudia este tema a través de las **Ideas clave** disponibles a continuación.

**E**n este tema veremos los **algoritmos de clasificación**, los cuales son un tipo de algoritmos de aprendizaje supervisado donde la variable a predecir es categórica.

- ▶ En primer lugar, se verán las diferencias entre un problema de clasificación binaria y un problema de clasificación multiclas. Un ejemplo de clasificador binario sería un algoritmo para determinar la probabilidad de que un mensaje de correo electrónico sea *spam* y no *spam*; o por ejemplo un algoritmo para predecir el impago de un crédito bancario. Por otro lado, un ejemplo de clasificador multiclas puede ser un algoritmo que busque predecir en qué país de doce posibles un cliente va a realizar la siguiente reserva.

Este tipo de algoritmos de aprendizaje supervisado tienen la ventaja competitiva de **poder adelantarte a situaciones futuras** de la forma más adecuadas y gestionarlas mejor.

Cualquier evaluación corre el riesgo de ser subjetiva, por tanto, es necesario establecer los criterios de evaluación más objetivos y rigurosos posibles. Por tanto, a continuación, se describen las diferentes métricas de evaluación de los algoritmos de clasificación más utilizadas.

## 3.2. Algoritmos de clasificación

**D**entro del **aprendizaje supervisado**, a continuación de los algoritmos de regresión, el siguiente gran grupo de **algoritmos** son los de **clasificación**. A diferencia de los algoritmos de regresión cuyo objetivo es predecir un valor numérico, los algoritmos de clasificación tienen como **objetivo obtener la clase más probable para cada una de las instancias**.

Este tipo de técnicas pueden utilizarse para predecir o estimar la probabilidad de pertenencia a **una clase de entre dos posibles**, lo que se conoce como **clasificación binaria**. Por otro lado, también se pueden utilizar para predecir o estimar la probabilidad de **pertenencia de una clase de entre varias posibles** (más de dos), lo cual se conoce como **clasificación multiclasa**.

A continuación, veremos las métricas de evaluación que se utilizan centrándonos en la matriz de confusión, precisión/recall, accuracy/sensitivity y f-measure. Posteriormente, veremos las **curvas ROC y la métrica de área bajo la curva**.

Dentro del **aprendizaje supervisado** los algoritmos de clasificación se utilizan para obtener la clase o categoría más probable de entre dos o más posibles.

## 3.3. Métricas de evaluación: matriz de confusión

**L**a mejor métrica de rendimiento de un algoritmo de clasificación es ver si el clasificador tiene **éxito para su propósito**. Para evaluar un clasificador se pueden utilizar los valores predichos de las clases, los valores reales de las clases o bien la probabilidad estimada de la predicción.

En la práctica lo habitual es **mantener dos vectores** de datos. Por un lado, **un vector que contiene la verdad o los valores observados y otro** vector que contiene **los valores predichos** o estimados. Es importante resaltar que ambos vectores deben tener el mismo tamaño y los datos en el mismo orden. En este tipo de algoritmos de aprendizaje supervisado la clase verdadera se conoce de antemano y esta variable se utiliza para evaluar el algoritmo en el conjunto de test comparándola con el resultado predicho.

Es decir, los valores predichos **se obtienen por medio de la aplicación de un modelo sobre el conjunto de test**, el cual ha sido entrenado previamente con datos de entrenamiento.

En la mayoría de los paquetes del lenguaje *R*, esto se realiza por medio de invocar a la función `predict()` sobre un objeto que es el resultado de un modelo entrenado previamente y sobre un conjunto o *data.frame* de test. Ejemplo:

```
pred_valores <- predict(my_model test_data)
```

La función `predict()` tiene parámetros para indicar el tipo de predicción, normalmente se utiliza el valor *prob* para obtener la probabilidad de pertenencia a cada una de las clases.

Incluso en el caso de los clasificadores binarios, que realizan la predicción de una clase en función de dos posibles categorías, es muy útil conocer con qué certeza o confianza se predice cada una de las clases.

Por ejemplo, un mensaje de correo electrónico puede ser predicho como *spam* con una probabilidad de 0,9 o de 0,59, etc. De forma general si dos algoritmos de clasificación diferentes producen los mismos errores, pero uno de ellos es más capaz de tener en cuenta la incertidumbre, sería un mejor modelo. **Por tanto, para evaluar correctamente el resultado de un clasificador es necesario considerar el valor de probabilidad obtenido** en lugar de utilizar únicamente la clase más probable.

En resumen, el objetivo es obtener modelos que tienen mucha confianza en las predicciones correctas y sean temerosos en las predicciones dudosas. Este balance entre confianza y prudencia es la clave de la evaluación de modelos.

#### Ejemplo. Clasificación binaria

En la siguiente tabla se muestra la distribución de probabilidad obtenida para cada una de las dos clases en un problema de clasificación binaria. En este caso es importante tener claro que indica la clase yes y la clase no, por ejemplo, la clase yes puede indicar que el cliente ha realizado un impago de su crédito o por el contrario que el cliente ha realizado un pago con éxito del crédito. Por regla general, a la clase positiva (yes o 1 en este caso) se hace corresponder con el evento interesante a predecir.

Es importante tener claro la definición de cada una de las clases para no obtener conclusiones contradictorias.

```
> head(predicted_prob)
      no      yes
1 0.0808272 0.9191728
2 1.0000000 0.0000000
3 0.7064238 0.2935762
4 0.1962657 0.8037343
5 0.8249874 0.1750126
6 1.0000000 0.0000000
```

Figura 1. Fuente: Brett, 2013.

Los valores anteriores son el resultado de la predicción, pero al tratarse de una clasificación binaria se puede obtener solo uno de ellos, pues la suma de los dos valores debe ser 1. A partir de esta probabilidad y aplicando un umbral o punto de corte a partir del cual se establece que la predicción es de una clase o de otra, se pueden realizar las evaluaciones correspondientes.

Por ejemplo, en la siguiente tabla se muestran los resultados obtenidos de un modelo de clasificación binaria donde la variable a predecir es si un correo electrónico es *spam* o no. El primer ejemplo se ha predicho como *ham* puesto que tiene una probabilidad de *spam* muy baja y en realidad era *ham*. El quinto ejemplo se ha predicho como *spam* con una certeza muy alta pues tiene una probabilidad de 1 y en realidad era *spam*.

```
> head(sms_results)
  actual_type predict_type    prob_spam
1      ham        ham 2.560231e-07
2      ham        ham 1.309835e-04
3      ham        ham 8.089713e-05
4      ham        ham 1.396505e-04
5    spam        spam 1.000000e+00
6      ham        ham 3.504181e-03
```

Figura 2. Fuente: Brett, 2013.

En el ejemplo anterior, cuando los valores predichos son de la clase *ham*, los valores de *spam*, son muy cercanos a 0 y por el contrario cuando los valores predichos son *spam* este valor es 1. Este comportamiento sugiere que el modelo tiene mucha confianza en sus clasificaciones.

En cualquier caso, lo habitual es que los **clasificadores o los modelos tengan errores y sus predicciones difieran del valor real**, como por ejemplo en la siguiente tabla donde todas las predicciones han sido *ham* y el valor real era *spam*.

```
> head(subset(sms_results, actual_type != predict_type))
   actual_type predict_type    prob_spam
53      spam        ham 0.0006796225
59      spam        ham 0.1333961018
73      spam        ham 0.3582665350
76      spam        ham 0.1224625535
81      spam        ham 0.0224863219
184     spam        ham 0.0320059616
```

Figura 3. Fuente: Brett, 2013.

Sin embargo, en algunos casos las probabilidades con las que se ha asignado la clase predicha no son muy extremas. Por ejemplo, la instancia 73 de la tabla anterior ha sido asignada a la clase *ham*, pero tenía una probabilidad de 0,35 o 35 % de ser *spam*.

Estos ejemplos muestran errores puntuales o particulares, pero para saber si un clasificador es útil se deben tener en cuenta métricas más completas sobre el conjunto de datos completo.

En el caso de los algoritmos de clasificación una métrica muy común es la matriz de confusión. Una matriz de confusión es una tabla que organiza las predicciones en función de los valores reales de los datos. Una de las dimensiones de la tabla hace referencia a la categoría de los valores predichos, la otra dimensión a las categorías reales.

Las instancias clasificadas correctamente caen en la diagonal de la matriz. Los valores fuera de la diagonal indican las instancias clasificadas incorrectamente, se trata de predicciones incorrectas. Las métricas de rendimiento obtenidas con la matriz de confusión se basan en cuantas instancias caen dentro y fuera de la diagonal. La mayoría de las métricas de rendimiento consideran la capacidad que tiene un clasificador de discernir una categoría respecto de las demás. La categoría de interés se conoce como clase positiva, mientras que las otras categorías se conocen como clase negativa.

En la siguiente imagen se muestra cómo se obtienen los cuatro valores de una matriz de confusión para el caso de una clasificación binaria.

|                 |   | Condition<br>(as determined by "Gold standard")                                     |                                  |  |
|-----------------|---|---|----------------------------------|--|
|                 |   | Condition Positive  | Condition Negative               |  |
| Test<br>Outcome | Test<br>Outcome<br>Positive   | True Positive   | False Positive<br>(Type I error) | Positive predictive value =<br>$\frac{\sum \text{True Positive}}{\sum \text{Test Outcome Positive}}$ |
|                 | Test<br>Outcome<br>Negative   | False Negative<br>(Type II error)   | True Negative                    | Negative predictive value =<br>$\frac{\sum \text{True Negative}}{\sum \text{Test Outcome Negative}}$ |
|                 | Sensitivity =<br>$\frac{\sum \text{True Positive}}{\sum \text{Condition Positive}}$ | Specificity =<br>$\frac{\sum \text{True Negative}}{\sum \text{Condition Negative}}$ |                                  |  |

Tabla 1. Matriz de Confusión.

Fuente: [https://en.wikipedia.org/wiki/Confusion\\_matrix](https://en.wikipedia.org/wiki/Confusion_matrix)

Cada uno de estos cuatro valores se pueden **definir** de la siguiente forma:

1. **Tasa de verdaderos positivos (true positives):** se trata de las **clasificaciones correctas** de las instancias que corresponden a la **clase positiva**.
2. **Tasa de falsos positivos (false positives):** se trata de las **clasificaciones de la clase negativa** que han sido **incorrectamente clasificadas** como clase positiva.
3. **Tasa de verdaderos negativos (true negatives):** se trata de las **clasificaciones correctas** de las instancias que corresponden a la **clase negativa**.
4. **Tasa de falsos negativos (false negatives):** se trata de las **clasificaciones de la clase positiva** que han sido **incorrectamente clasificadas** como clase negativa.

## Matriz de Confusión en R

Una forma sencilla de obtener la matriz de confusión utilizando el lenguaje de programación *R* es por medio de la función `table()`, la cual devuelve las ocurrencias de cada combinación de valores, por ejemplo:

```
> table(sms_results$actual_type, sms_results$predict_type)
   ham spam
ham 1202    5
spam   29 154
```

Figura 4. Fuente: Brett, 2013.

Para obtener directamente las probabilidades se puede utilizar:

```
prop.table(table(sms_results$actual_type, sms_results$predict_type),
           margin = 2)
```

Por medio de la matriz de confusión se pueden obtener también las siguientes métricas:

- ▶ **Accuracy**: esta métrica también se conoce como el ratio de éxito. Representa la proporción del número de predicciones correctas entre el número total de predicciones y se define así:

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- ▶ **Sensitivity/Specificity** (sensibilidad/especificidad): una clasificación siempre conlleva un balance entre ser conservador y agresivo. Por ejemplo, un sistema de clasificación de *spam* podría eliminar todos los mensajes de *spam* a costa de eliminar también los mensajes *ham*.

Por otro lado, es necesario una garantía de que ningún mensaje *ham* sea clasificado como *spam*. Este balance se captura con las métricas de sensibilidad y especificidad.

La sensibilidad de un modelo es el ratio de los ejemplos positivos correctamente clasificados. La especificidad de un modelo indica la proporción de los ejemplos negativos correctamente clasificados.

Estas métricas tienen valores de 0 a 1, siendo 1 lo más deseable y se definen así:

$$\text{especificidad} = \frac{TN}{TN + FP}$$

$$\text{sensibilidad} = \frac{TP}{TP + FN}$$

- ▶ **Precision/recall**: estas métricas hacen referencia a los compromisos hechos en la clasificación. Se utilizan mucho en el campo de *information retrieval* e indican lo interesantes y relevantes que son los resultados de un modelo.

La métrica de precisión también se conoce como *positive predictive value (PPV)* e indica la proporción de ejemplos que son verdaderamente positivos. Es decir, cuando el modelo predice la clase positiva, ¿cuántas veces está en lo cierto? Por otro lado, la métrica de *recall* indica que tan completos son los resultados. La métrica de *recall* equivale a la métrica de sensibilidad. Un modelo con gran *recall* captura un gran porcentaje de ejemplos positivos. La definición de ambas métricas es la siguiente:

$$\text{precision} = \frac{TP}{TP + FP}$$

$$\text{recall} = \frac{TP}{TP + FN}$$

- ▶ **F-measure:** también conocida como *F1*, es una métrica que combina precisión y *recall* utilizando la media armónica. Se utiliza la media armónica porque los valores indican proporciones entre 0 y 1. Se utiliza con mucha frecuencia puesto que simplifica el rendimiento de un algoritmo de clasificación a una única métrica. De forma matemática se define así:

$$F1 = \frac{2 * precision * recall}{recall + precision} = \frac{2 * TP}{2 * TP + FP + FN}$$

La métrica por defecto asume que precisión y *recall* tienen la misma importancia, pero es posible ponderarlos para darle mayor peso a uno u otro

A la hora de evaluar los algoritmos de clasificación es importante diferenciar entre una clasificación binaria o multiclase. Una de las métricas más comunes es la matriz de confusión que nos proporciona información acerca de los aciertos y errores de cada una de las clases. A partir de la matriz de confusión se pueden obtener métricas como *accuracy*, *sensitivity/specifity*, *precisión/recall* y *f-measure*.

### 3.4. Métricas de evaluación: curvas ROC, AUC

**L**a clave de cualquier modelo de aprendizaje automático es su **capacidad de generalizar situaciones del futuro** en función de los datos históricos observados. Las métricas anteriores que se obtienen a partir de la matriz de confusión llevan que se establece un punto de corte determinado sobre la distribución de probabilidad para determinar si una observación es clasificada como una clase determinada. Es decir, por ejemplo, aquellos valores por encima de 0,5 se les asigna la clase positiva (1) y aquellos valores iguales o por debajo de 0,5 la clase negativa (0).

En el caso de los algoritmos de clasificación binaria, se puede utilizar una métrica obtenida a partir de las curvas *receiver operating characteristic (ROC)* conocida como *area under the curve (AUC)* o en castellano área bajo la curva. Esta métrica se utiliza para determinar el balance entre la detección de verdaderos positivos y evitar los falsos positivos. Para ello, se muestra la proporción de detección de los verdaderos positivos en el eje vertical y la proporción de los falsos positivos en el eje horizontal, para un umbral o punto de corte determinado.

Los puntos a lo largo de la curva indican el ratio de los verdaderos positivos a medida que se incrementan los valores de los falsos positivos. En la siguiente imagen se muestra un ejemplo de cuatro curvas ROC distintas.

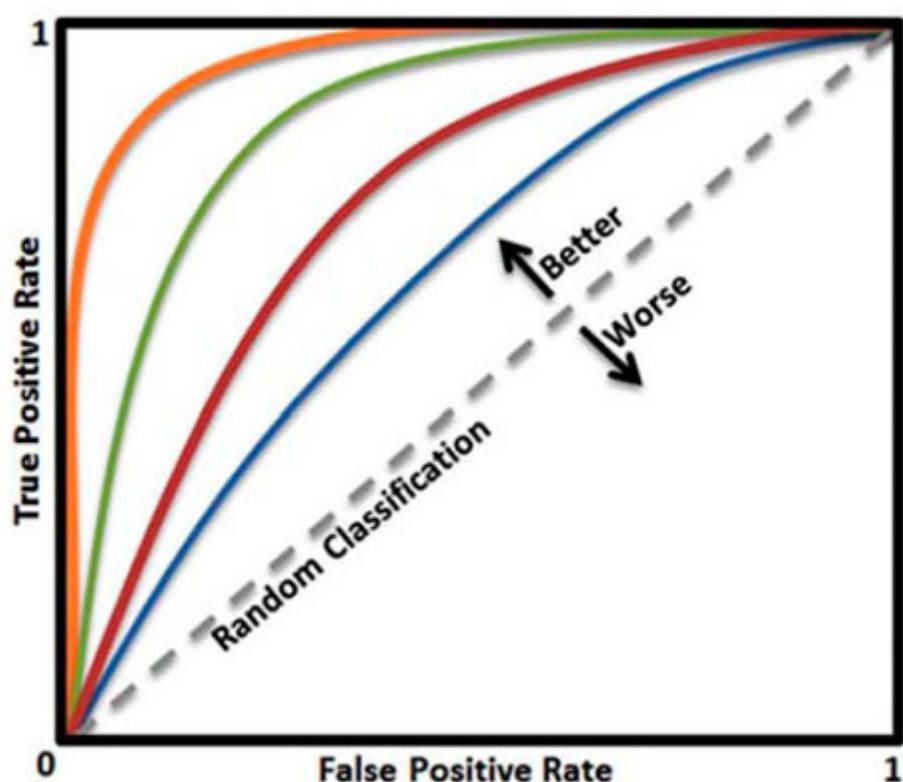


Figura 5. Ejemplo de una curva ROC.

Recuperado de: [https://openi.nlm.nih.gov/imgs/512/261/3861891/PMC3861891\\_CG-14-397\\_F10.png](https://openi.nlm.nih.gov/imgs/512/261/3861891/PMC3861891_CG-14-397_F10.png)

A medida que la curva este más cercana de la esquina superior izquierda, mejor será.

En el ejemplo, el clasificador de la curva naranja es mejor que el verde, y este mejor que el rojo y el azul. Una clasificación completamente aleatoria coincide con la línea punteada.

A partir de estas curvas se puede obtener el **área bajo la curva**. Esta métrica va desde valores de 0,5 para un clasificador sin potencia predictiva (completamente aleatorio) hasta 1 para un clasificador perfecto. Cuanto más cercanos están los resultados del clasificador perfecto, mejor.

Es posible que para un mismo valor de AUC haya diferentes curvas ROC, por lo que suele ser buena práctica mostrarlas de forma gráfica.

## Visualizando el rendimiento en R

El paquete **ROCR** (<https://rocr.bioinf.mpi-sb.mpg.de/>) proporciona una serie de funciones para visualizar y entender el rendimiento de un clasificador.

Para crear visualizaciones con ROCR se necesitan dos vectores: uno con la clase real de los valores predichos y otro con la probabilidad estimada de la clase positiva. Estos valores se utilizan en la función `prediction()` que devuelve un objeto que se puede utilizar para obtener métricas o visualizar el resultado. El siguiente código de ejemplo muestra el uso:

```

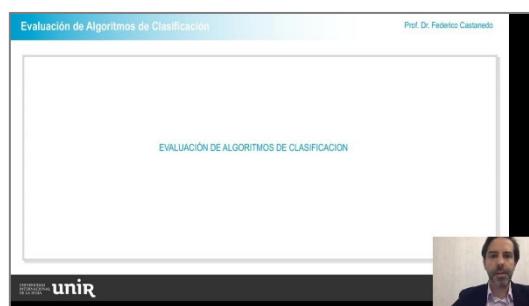
library(ROCR)
pred      <-      prediction(predictions      =
pred_results$prob,
                                labels = pred_results$actual_value)
#Curva ROC
perf <- perfomance(pred, "tpr", "fpr")
plot(perf)

#Curva Precision-Recall
perf2 <- performance(pred, "prec", "rec")
plot(perf2)

#Curva sensivity-specificity
perf3 <- performance(pred, "sens", "spec")
plot(perf3)

```

En este vídeo se van a comentar y poner de relieve los principales métodos existentes para la evaluación de los modelos de clasificación y por qué es importante realizar esta evaluación.



Evaluación de algoritmos de clasificación.

---

Accede al vídeo a través del aula virtual

---

# Lo + recomendado

## No dejes de leer

### Performance Measures for Machine Learning

Cornell CIS. Computer Science. (s. f.) Performance Measures for Machine Learning.

Presentación de la Universidad de Cornell sobre métricas de rendimiento de los modelos de aprendizaje automático.

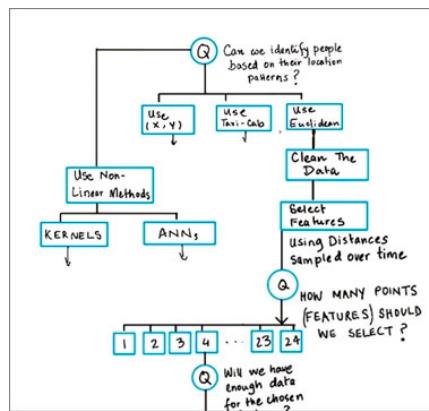
Accede al artículo a través del aula virtual o desde la siguiente dirección web:

[https://www.cs.cornell.edu/courses/cs578/2003fa/performance\\_measures.pdf](https://www.cs.cornell.edu/courses/cs578/2003fa/performance_measures.pdf)

## No dejes de ver

### Performance Metrics Precision, Recall and F Score – Model Building and Validation

Métricas de rendimiento de los modelos de aprendizaje automático.



Accede al vídeo a través del aula virtual o desde la siguiente dirección web:

<https://www.youtube.com/watch?v=uZqwmihvTFY>

### Machine Learning: Testing and Error Metrics

Vídeo Tutorial sobre métricas de error y test.

### Machine learning: Testing and Error Metrics

Accede al vídeo a través del aula virtual o desde la siguiente dirección web:

<https://www.youtube.com/watch?v=aDW44NPhNw0>

## A fondo

### Top 15 Evaluation Metrics for Classification

Machine Learning Plus (30 septiembre de 2017). Top 15 Evaluation Metrics for Classification [mensaje en un blog]

Descripción de las 15 métricas más importantes para la evaluación de algoritmos.

Accede al artículo a través del aula virtual o desde la siguiente dirección web:

<https://www.machinelearningplus.com/evaluation-metrics-classification-models-r/>

## Webgrafía

### Scikit Learn

Página web que describe ejemplos de *precision* y *recall* en Python.



Accede a la página web a través del aula virtual o desde la siguiente dirección web:

[http://scikit-learn.org/stable/auto\\_examples/model\\_selection/plot\\_precision\\_recall.html](http://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html)

## Bibliografía

Brett, L. (2013). *Machine Learning with R*. Birmingham: Packt.

1. Los algoritmos de clasificación tienen como objetivo:
  - A. Predecir la clase más probable de entre varias posibles.
  - B. Predecir la distribución de probabilidad de las clases de cada instancia.
  - C. Predecir un valor numérico como variable objetivo.
  
2. Dado dos clasificadores binarios si se equivocan en las clases más probables de las mismas instancias
  - A. Son igual de buenos.
  - B. Si uno tiene una mayor incertidumbre que el otro es peor clasificador.
  - C. Ninguna de las anteriores.
  
3. Cuáles de las siguientes métricas son utilizadas en los modelos de clasificación:
  - A. Precisión.
  - B. *Recall*.
  - C. MSE.
  - D. *F-measure*.
  
4. Un matriz de confusión es:
  - A. Una tabla que organiza las predicciones en función de los valores reales de los datos.
  - B. Una tabla que tiene las predicciones de los diferentes clasificadores.
  - C. Una tabla que contiene el valor real y el valor deseado.

5. La métrica de *accuracy*:

- A. Se conoce como la ratio de éxito.
- B. Representa el número de predicciones correctas entre el total de predicciones.
- C. Representa el número de falsos detectados.
- D. Ninguna de las anteriores es correcta.

6. Un modelo con gran valor de precisión:

- A. Indica que la mayoría de las veces que se predice la clase negativa está en lo cierto.
- B. Indica que la mayoría de las veces que se predice la clase positiva está en lo cierto.
- C. Las dos anteriores son correctas.

7. Un modelo con gran valor de *recall*:

- A. Captura un gran porcentaje de ejemplos negativos.
- B. Captura un gran porcentaje de ejemplos positivos.
- C. Las dos anteriores son correctas.

8. La métrica *F1*:

- A. Se trata de una forma de calcular el AUC.
- B. Combina precisión y *recall*.
- C. Combina sensibilidad y especificidad.

9. El área bajo la curva (AUC):

- A. Es un valor comprendido entre 0 y 1.
- B. Es un valor comprendido entre 0 y 100.
- C. Es un porcentaje entre 0 y 100 %.

**10.** La curva ROC mide:

- A. La ratio de verdaderos positivos y falsos positivos.
- B. La ratio de verdaderos positivos y falsos negativos.
- C. La ratio de verdaderos negativos y falsos positivos.

Aprendizaje Automático

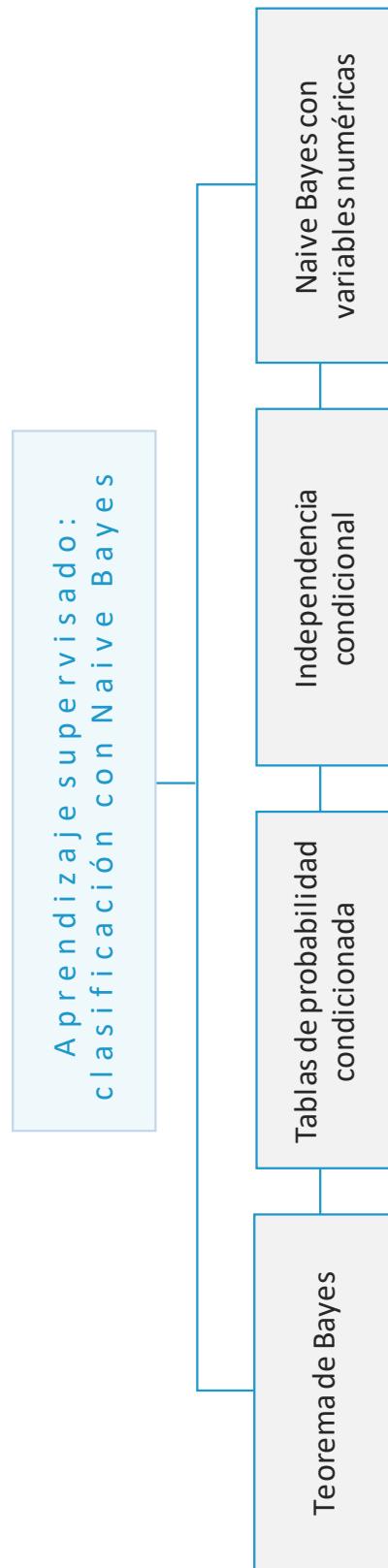
---

# Aprendizaje supervisado: clasificación con Naive Bayes

# Índice

|   |           |
|---|-----------|
| <b>Esquema</b>  | <b>3</b>  |
| <b>Ideas clave</b>                                    | <b>4</b>  |
| 4.1. ¿Cómo estudiar este tema?                        | 4         |
| 4.2. Teorema de Bayes                                 | 4         |
| 4.3. Tablas de probabilidad condicionada              | 7         |
| 4.4. Independencia condicional en el clasificador     |           |
| Naive Bayes   | 10        |
| 4.5. Clasificador Naive Bayes                         | 11        |
| 4.6. Clasificador Naive Bayes con variables numéricas |           |
| 13  | 13        |
| 4.7. Referencias bibliográficas                       | 15        |
| <b>Lo + recomendado</b>                               | <b>16</b> |
| <b>+ Información</b>                                  | <b>19</b> |
| <b>Test</b>   | <b>20</b> |

# Esquema



## 4.1. ¿Cómo estudiar este tema?

Estudia este tema a través de las **Ideas clave** disponibles a continuación.

**E**n este tema se va a introducir el clasificador Naive Bayes, el cual está basado en el teorema de Bayes para calcular las probabilidades *a posteriori* de los eventos a predecir.

Vamos a ver los siguientes puntos:

- ▶ En primer lugar, veremos el teorema de Bayes.
- ▶ A continuación, se describe la forma de calcular las tablas de probabilidad condicionada.
- ▶ Posteriormente se verá por qué es importante asumir independencia condicional en el clasificador Naive Bayes.
- ▶ El clasificador Naive Bayes y finalmente cómo se puede utilizar con variables numéricas.

## 4.2. Teorema de Bayes

**E**l Teorema de Bayes es una proposición planteada por el filósofo inglés Thomas Bayes (1702-1761) en el año 1763 en su artículo «An Essay towards solving a Problem in the Doctrine of Chances» publicado en la revista *Philosophical Transactions of the Royal Society of London*.

Este teorema expresa la **probabilidad condicional** de un evento aleatorio A dado B en términos de la distribución de probabilidad condicional del evento B dado A y la distribución de probabilidad marginal de solo A.



Foto: Reverendo Thomas Bayes.  
Recuperado de: [https://upload.wikimedia.org/wikipedia/commons/d/d4/Thomas\\_Bayes.gif](https://upload.wikimedia.org/wikipedia/commons/d/d4/Thomas_Bayes.gif)

El teorema de Bayes es bastante relevante porque **relaciona la probabilidad de dos eventos A y B utilizando la dependencia condicional de uno de ellos**. Es decir, relaciona la probabilidad de que ocurra el evento A y sabemos de antemano que ha ocurrido B, utilizando la probabilidad de que ocurra el evento B sabiendo que ha ocurrido A.

Este teorema **permite relacionar, entre otras cosas, síntomas y enfermedades**. Por ejemplo, sabiendo la probabilidad de tener dolor de garganta dado que se conoce que se tiene gripe, se puede obtener la probabilidad de tener gripe dado que se tiene dolor de garganta.

El teorema de Bayes **relaciona la comprensión de la probabilidad de aspectos causa-efecto dados los eventos dependientes observados**. Un evento dependiente es aquel cuyo resultado se ve afectado por el resultado de otro evento o serie de eventos. Los **eventos dependientes** son la base del modelado predictivo puesto que se busca obtener la probabilidad de que ocurra un suceso teniendo en cuenta la existencia de una serie de eventos dependientes.

En el caso de dos eventos dependientes A y B, el teorema de Bayes describe su relación de la siguiente manera:

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

Es decir, la probabilidad de A dado que ocurre B, es igual a la probabilidad de que ocurra B dado que ha ocurrido A, multiplicado por la probabilidad de que ocurra A, dividido por la probabilidad de que ocurra B.

Este teorema se puede generalizar para más de dos eventos de la siguiente forma:

en primer lugar, se entiende por evento mutuamente excluyente cuando dos resultados diferentes de un evento no pueden ocurrir al mismo tiempo. En el caso además de que los eventos sean exhaustivos, por lo menos uno de ellos tiene que ocurrir.

De forma matemática, se puede definir el teorema de Bayes para  $n$  eventos: Sea  $\{A_1, A_2, \dots, A_i, \dots, A_n\}$  un conjunto de sucesos mutuamente excluyentes y exhaustivos, y tales que la probabilidad de cada uno de ellos es distinta de cero. Sea B un suceso cualquiera del que se conocen las probabilidades condicionales  $P(B|A_i)$ . Entonces, la probabilidad  $P(A_i|B)$  viene dada por la expresión:

$$P(A_i|B) = \frac{P(B|A_i) * P(A_i)}{P(B)}$$

Donde:

- ▶  $P(A_i)$  son las probabilidades *a priori*.
- ▶  $P(B|A_i)$  es la probabilidad de B en la hipótesis  $A_i$ .
- ▶  $P(A_i|B)$  son las probabilidades *a posteriori*.
- ▶  $P(B)$  es la verosimilitud marginal.

### Ejemplo:

Supongamos que deseamos estimar la probabilidad de que un mensaje de correo electrónico sea *spam*. Sin tener evidencias adicionales y, únicamente, por los mensajes previos obtenidos la probabilidad *a priori* de que un mensaje sea *spam* es 0,2.

Ahora bien, si tenemos evidencia de que el mensaje contiene la palabra Viagra, por medio de conocer la probabilidad de que la palabra Viagra haya sido utilizada en mensajes de *spam* previos (lo cual se conoce como verosimilitud o *likelihood*) y por medio de la probabilidad de que la palabra Viagra aparezca en cualquier mensaje, lo que se conoce como verosimilitud marginal. Aplicando el teorema de Bayes se puede calcular la probabilidad *a posteriori* y si esta es mayor que 0,5 es más probable que el mensaje sea *spam*. En la siguiente formula se muestra el cálculo anteriormente descrito.

$$P(Spam_t|Viagra) = \frac{P(Viagra|Spam) * P(Spam)}{P(Viagra)}$$

El **teorema de Bayes** relaciona la probabilidad de dos o más eventos utilizando la dependencia condicional de cada uno de ellos. Los eventos deben ser dependientes y mutuamente excluyentes.

## 4.3. Tablas de probabilidad condicionada

Para obtener cada uno de los componentes de las formulas anteriores es necesario construir una **tabla de frecuencias** que indica el número de veces que el evento aparece en cada una de las situaciones. En nuestro ejemplo de *spam*, es necesario calcular el número de veces que la palabra Viagra ha aparecido en los mensajes de

*spam*. Esta tabla de frecuencias se utiliza posteriormente para calcular las tablas de verosimilitud o de probabilidad condicionada.

Siguiendo con el ejemplo de los mensajes de *spam*, en el caso hipotético de que tuviéramos la siguiente distribución histórica de 100 mensajes para la palabra Viagra...

| Frecuencia  | Si | No | Total |
|-------------|----|----|-------|
| <i>Spam</i> | 4  | 16 | 20    |
| <i>Ham</i>  | 1  | 79 | 80    |
| Total       | 5  | 95 | 100   |

Tabla 1. Ejemplo de distribución histórica de mensajes *spam* y *ham* para la palabra Viagra.

...Obtendríamos la correspondiente **tabla de verosimilitud**:

| Verosimilitud | Si    | No     | Total |
|---------------|-------|--------|-------|
| <i>Spam</i>   | 4/20  | 16/20  | 20    |
| <i>Ham</i>    | 1/80  | 79/80  | 80    |
| Total         | 5/100 | 95/100 | 100   |

Tabla 2. Ejemplo de tabla de verosimilitud para mensajes *spam* y *ham* en función de la palabra Viagra.

Con estos datos, para calcular la probabilidad *a posteriori* de que un mensaje sea *spam* dado que nos ha llegado la palabra Viagra, tendríamos que hacer el siguiente cálculo:

$$P(\text{Spam}|\text{Viagra}) = [(4/20) * (20/100)] / (5/100) = 0.8$$

Es decir, con los datos anteriores, la probabilidad de que un correo electrónico que contenga la palabra Viagra sea *spam* es del 0,8.

Ahora supongamos que deseamos añadir a este cálculo otros términos más comunes que aparecen en los mensajes *spam*, como pueden ser: *money*, *groceries* y *unsubscribe*.

En este caso, tendríamos la siguiente tabla de verosimilitud:

|               | Viagra (W1) |       | Money (W2) |       | Groceries (W3) |        | Unsubscribe (W4) |        |       |
|---------------|-------------|-------|------------|-------|----------------|--------|------------------|--------|-------|
| Verosimilitud | Si          | No    | Si         | No    | Si             | No     | Si               | No     | Total |
| <i>Spam</i>   | 4/20        | 16/20 | 10/20      | 10/20 | 0/20           | 20/20  | 12/20            | 8/20   | 20    |
| <i>Ham</i>    | 1/80        | 79/80 | 14/80      | 66/80 | 8/80           | 71/80  | 23/80            | 57/80  | 80    |
| Total         | 5/10        | 95/10 | 24/10      | 76/10 | 8/100          | 91/100 | 35/100           | 65/100 | 100   |

Tabla3. Ejemplo de tabla de verosimilitud para los mensajes *spam* y *ham* en función de las palabras Viagra, money, groceries y unsubscribe.

De esta forma, si llega un nuevo mensaje que contiene las palabras Viagra y *unsubscribe*, pero no *money* ni *groceries*; utilizando el teorema de Bayes habría que calcular la siguiente formula:

$$P(\text{Spam}|\text{Viagra}) = \frac{P(\text{Viagra}|\text{Spam}) * P(\text{Spam})}{P(\text{Viagra})}$$

El cálculo de la fórmula anterior es **computacionalmente costoso**, puesto que a medida que se añaden nuevos términos son necesarias grandes cantidades de memoria para almacenar todas las combinaciones.

## 4.4. Independencia condicional en el clasificador

### Naive Bayes

**D**ebido a que el cálculo riguroso de la fórmula del teorema de Bayes, como en el ejemplo anterior, es computacionalmente costoso, el clasificador Naive Bayes se basa en una **modificación sencilla**. Básicamente asume **independencia condicional** entre los eventos, a pesar de que si todos los eventos fueran independientes sería imposible predecir ningún evento con los datos observados por otro. Formalmente, **dos eventos son independientes** si el resultado del segundo evento no es afectado por el resultado del primer evento. Si A y B son eventos independientes, la probabilidad de que ambos eventos ocurran es el producto de las probabilidades de los eventos individuales.

$$P(A|B) = P(B|A) = P(B) * P(A)$$

Por otro lado, los **eventos dependientes son la base del modelado predictivo**, puesto que permiten predecir la presencia de un evento en función de otro. Por ejemplo, la presencia de nubes suele ser un evento predictivo de un día lluvioso, o la presencia de la palabra *viagra* en un correo electrónico suele ser un evento predictivo de *spam*.

No obstante, al no poder asumir dependencia condicional por el alto coste computacional, el clasificador Naive Bayes asume **independencia condicional** entre los eventos condicionados al mismo valor de la clase. Este hecho es el que le ha dado el adjetivo de *naive* al clasificador.

En nuestro ejemplo anterior, asumiendo independencia condicional de las palabras para obtener la probabilidad de *spam*, tendríamos la siguiente formula:

$$\begin{aligned} P(\text{Spam} | W_1 \cap \neg W_2 \cap \neg W_3 \cap W_4) \\ = \frac{P(W_1|\text{Spam})P(\neg W_2|\text{Spam})P(\neg W_3|\text{Spam})P(W_4|\text{Spam}) * P(\text{Spam})}{P(W_1)P(\neg W_2)P(\neg W_3)P(W_4)} \\ = (4/20) * (10/20) * (20/20) * (12/20) * (20/100) = 0.012 \end{aligned}$$

Por otro lado, para obtener la probabilidad de *ham*, tendríamos:

$$\begin{aligned} P(Ham|W_1 \cap \neg W_2 \cap \neg W_3 \cap W_4) \\ = \frac{P(W_1|Ham)P(\neg W_2|Ham)P(\neg W_3|Ham)P(W_4|Ham) * P(Ham)}{P(W_1)P(\neg W_2)P(\neg W_3)P(W_4))} \\ = (1/80) * (66/80) * (71/80) * (23/80) * (80/100) = 0.002 \end{aligned}$$

Como  $0,012/0,002 = 6$ , se puede afirmar que es 6 veces más probable que el mensaje sea *spam* que *ham*.

Si queremos calcular la probabilidad de que el mensaje sea *spam*, sería igual a la verosimilitud de que el mensaje sea *spam* dividido por la verosimilitud de que sea *spam* o *ham*:  $0,012 / (0,012 + 0,002) = 0,857$

Análogamente, la probabilidad de ser *ham* es:  $0,002 / (0,002 + 0,012) = 0,143$

Por tanto, podemos estimar que, dadas las palabras del mensaje, hay una probabilidad de 0,857 de que sea *spam* y de 0,143 de que sea *ham* y como son eventos mutuamente excluyentes suman 1.

## 4.5. Clasificador Naive Bayes

Como hemos comentado previamente, el teorema de Bayes es la **base** para el clasificador Naive Bayes. Este clasificador utiliza las probabilidades *a priori* de los eventos para estimar la **probabilidad de eventos futuros** por medio del teorema de Bayes.

Este clasificador utiliza datos históricos o de entrenamiento para calcular la probabilidad observada de cada evento en función de su vector de características.

Para realizar una predicción, el clasificador es utilizado con datos que tienen la clase desconocida y se utilizan las probabilidades observadas para estimar la clase más probable.

La fórmula general del clasificador Naive Bayes se puede definir de la siguiente manera: la probabilidad del nivel  $L$  de la clase  $C$ , dada la evidencia proporcionada por las variables de  $F_1, \dots, F_n$ , es igual al producto de las probabilidades de cada evidencia condicionada al nivel de la clase, la probabilidad *a priori* del nivel de la clase y un factor de escala  $1/Z$  que convierte el resultado en probabilidad:

$$P(C_L|F_1, \dots, F_n) = \frac{1}{Z} p(C_L) \prod_{i=1}^n p(F_i|C_L)$$

Este clasificador se utiliza principalmente para **clasificar texto, para detección de intrusos en redes de computadores, diagnósticos médicos, etc.** Por ejemplo, se puede utilizar la frecuencia de las palabras de los correos electrónicos para identificar nuevos correos *spam* en el futuro.

### Combinaciones desconocidas

Supongamos que ahora recibimos un mensaje que contiene las palabras *Viagra*, *groceries*, *money* y *unsubscribe*, y queremos estimar la probabilidad de que el mensaje sea *Viagra*. En este caso la verosimilitud de *spam* es:

$$(4/20) * (10/20) * (0/20) * (12/20) * (20/100) = 0$$

Por otro lado, la verosimilitud de *ham* es:

$$(1/80) * (14/80) * (8/80) * (23/80) * (80/100) = 0.00005$$

La probabilidad de *spam* es:

$$0/(0 + 0.0099) = 0$$

Y la probabilidad de *ham* es:

$$0.00005 / (0 + 0.00005) = 1$$

Este problema sucede cuando un evento nunca ha ocurrido para una o más categorías de las clases. Por ejemplo, si nunca se ha visto el término *groceries* en un mensaje *spam*  $P(\text{Spam} / \text{groceries}) = 0$ .

La solución es añadir un pequeño número a todas las clases en la tabla, para asegurarse que no existe ninguna combinación con probabilidad de ocurrir igual a 0, esto se conoce con el nombre de **estimador de Laplace**.

Por ejemplo, si usamos un valor de 1, la verosimilitud de *spam* y *ham* quedaría:

$$(5/24) * (11/24) * (1/24) * (13/24) * (20/100) = 0.0004$$
$$(2/84) * (15/84) * (9/84) * (24/84) * (80/100) = 0.0001$$

Lo que indica que la probabilidad de que el mensaje sea *spam* es del 0,8 y de que sea *ham* del 0,2.

## 4.6. Clasificador Naive Bayes con variables numéricas

Debido a que el clasificador Naive Bayes utiliza tablas de frecuencias para calcular las probabilidades, cada una de las variables utilizada debe de ser **categórica** y no se pueden utilizar de forma directa variables numéricas.

Una solución sencilla es **discretizar las variables numéricas en N conjuntos, agrupamientos o bins**. Este método es ideal cuando hay grandes cantidades de datos. Una cuestión importante aquí es considerar el punto de corte óptimo para hacer cada uno de los agrupamientos. Una buena solución suele ser explorar los datos para observar los puntos de corte en la distribución de los datos.

Por ejemplo, el siguiente histograma:

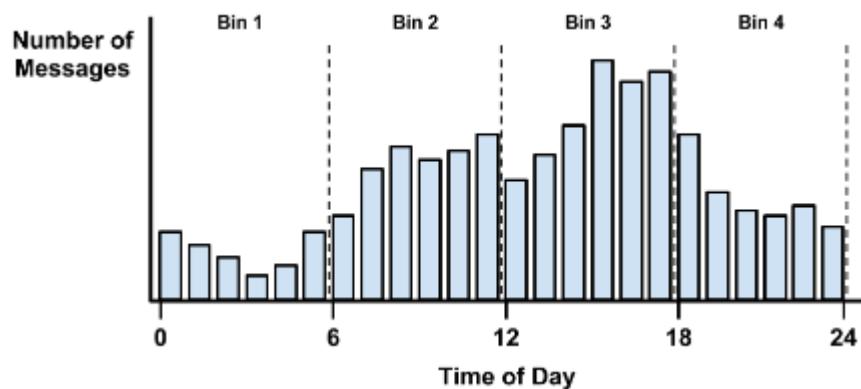
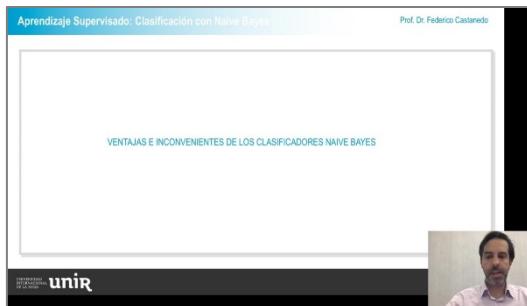


Gráfico 1. Fuente: Lantz, 2013.

Sugiere realizar una división en cuatro *bins*.

La **discretización** siempre se traduce en una reducción de la información, ya que la **granularidad inicial se reduce**. Por tanto, es importante mantener un balance entre el número de *bins*, puesto que con muy pocas se pierda mucha información y con muchas el proceso es muy costoso.

En el siguiente vídeo se van a comentar las ventajas e inconvenientes a la hora de utilizar un modelo de clasificación basado en Naive Bayes.



Ventajas e inconvenientes de los clasificadores Naive Bayes.

---

Accede al vídeo a través del aula virtual

---

A continuación, se muestra un ejemplo de cómo entrenar un clasificador Naive Bayes utilizando el lenguaje de programación R.

A screenshot of the RStudio IDE. The left pane shows an R script named 'Naive\_Bayes.R' with the following code:

```
1 #####  
2 # Ejemplo Clasificador Naïve Bayes #  
3 #####  
4  
5 install.packages("tm")  
6 library(tm)  
7 #install.packages("e1071")  
8 library(e1071)  
9 # Vamos a utilizar un dataset de mensajes SMS con 5575 instancias  
10 # de las cuales 4831 instancias son ham y 747 spam  
11 #####  
12 # 1. Preparación de datos #  
13 #####  
14  
15 # Vamos a utilizar una representación bag-of-words que ignora el orden  
16 # en que aparecen las palabras y proporciona una variable indicando  
17 # si aparece  
18  
19 sms_data <- read.csv("/Users/fcastan/Desktop/stuff/Clases/UNIR/Videos/NB_example/sms_data.csv",  
20 stringsAsFactors = FALSE)  
21  
22 # observamos el formato del dataset  
23 head(sms_data)  
24 tail(sms_data)  
25
```

The right pane shows the R console with the message 'Environment is empty'. Below the RStudio window, the text 'Tutorial Clasificador Naïve Bayes.' is visible.

Tutorial Clasificador Naïve Bayes.

---

Accede al vídeo a través del aula virtual

---

## 4.7. Referencias bibliográficas

Lantz, B. (2013). *Machine Learning with R*. Packt

# Lo + recomendado

## No dejes de leer

### Naive Bayes

Scikit learn. (s.f.). Naive Bayes.

Ejemplo de creación de un modelo Naive Bayes utilizando la librería scikit-learn de Python.

Accede al artículo a través del aula virtual o desde la siguiente dirección web:

[http://scikit-learn.org/stable/modules/naive\\_bayes.html](http://scikit-learn.org/stable/modules/naive_bayes.html)

### How to Implement Naive Bayes From Scratch in Python

Brownlee, J. (diciembre, 2014). How to Implement Naive Bayes From Scratch in Python. *machinelearningmastery.com*.

Ejemplo de creación de un modelo de Naive Bayes desde cero utilizando el lenguaje de programación Python.

Accede al artículo a través del aula virtual o desde la siguiente dirección web:

<https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/>

## Naive Bayes en Python

Greg. (11 de diciembre de 2014). Naive Bayes en Python [Mensaje en un blog]. *The Yat Blog.*

Explicación de cómo funciona el algoritmo Naive Bayes y ejemplo de implementación utilizando el lenguaje Python.

---

Accede al artículo a través del aula virtual o desde la siguiente dirección web:

<http://blog.yhat.com/posts/naive-bayes-in-python.html>

---

## Doing Naive Bayes Classification: aplicado al libro *50 sombras de Grey*

Cherny, L. (octubre de 2015). Doing Naive Classification.

Ejemplo de un clasificador Naive Bayes sobre el texto de las páginas del libro *50 sombras de Grey* con el objetivo de clasificar el contenido de las páginas.

---

Accede al artículo a través del aula virtual o desde la siguiente dirección web:

[http://nbviewer.jupyter.org/github/arnicas/NLP-in-Python/blob/master/4.%20Naive%20Bayes%20Classification.ipynb?imm\\_mid=0cd660&cmp=em-data-na-na-newsltr\\_20150225](http://nbviewer.jupyter.org/github/arnicas/NLP-in-Python/blob/master/4.%20Naive%20Bayes%20Classification.ipynb?imm_mid=0cd660&cmp=em-data-na-na-newsltr_20150225)

---

## No dejes de ver

### Clasificadores Naïve Bayes

Breve introducción al funcionamiento de los clasificadores Naïve Bayes.

## Clasificadores Naïve Bayes

---

Accede al vídeo a través del aula virtual o desde la siguiente dirección web:

<https://www.youtube.com/watch?v=oQ1OyqvL7dQ>

---

## Bibliografía

Lantz, B. (2013) *Machine Learning with R*. Birmingham: Packt

**1.** El teorema de Bayes:

- A. Fue propuesto por el reverendo Thomas Bayes.
- B. Relaciona la probabilidad de dos eventos A y B utilizando la dependencia condicional de uno de ellos.
- C. Relaciona la probabilidad de dos eventos A y B utilizando la dependencia condicional de ambos de ellos.

**2.** En el teorema de Bayes:

- A. Los eventos deben ser dependientes y mutuamente excluyentes.
- B. Los eventos deben ser independientes y mutuamente excluyentes.
- C. Los eventos deben ser independientes.

**3.** Si dos eventos son exhaustivos:

- A. Deben ocurrir los dos.
- B. Al menos debe ocurrir uno de ellos.
- C. Ninguna de las anteriores es correcta.

**4.** Dos eventos son mutuamente excluyente:

- A. Cuando siempre debe ocurrir el mismo evento.
- B. Cuando dos resultados diferentes de un mismo evento no pueden ocurrir al mismo tiempo.
- C. Ninguna de las anteriores es correcta.

**5.** Las tablas de frecuencias:

- A. Indican el número de veces que el evento aparece en cada una de las situaciones.
- B. Sirven para medir el éxito del modelo.
- C. Son la base para la construcción del modelo Naive Bayes.

**6.** Los eventos dependientes:

- A. Permiten estimar la presencia de un evento en función del otro.
- B. Siempre ocurren a la vez.
- C. Implica que la existencia de uno puede conllevar la existencia del otro.

**7.** ¿Cuáles de las siguientes afirmaciones son ciertas sobre el clasificador de Naive Bayes?

- A. Utiliza datos históricos para obtener la probabilidad observada de cada evento en función de su vector de características.
- B. Asume independencia condicional entre los eventos.
- C. El cálculo riguroso del teorema de Bayes es computacionalmente costoso.

**8.** Cuando existen combinaciones desconocidas en los datos de entrada:

- A. Las probabilidades *a posteriori* obtenidas pueden no tener sentido.
- B. El teorema de Bayes utiliza el estimador de Laplace.
- C. Se eliminan estas combinaciones de los datos de entrada.

**9.** La discretización de variables:

- A. Es una técnica que se aplica para utilizar el clasificador Naive Bayes con variables numéricas.
- B. Es ideal cuando hay grandes cantidades de datos.
- C. Funciona mejor cuando hay pocos datos.

**10.** La discretización de variables:

- A. Siempre se traduce en reducción de información.
- B. Nunca se traduce en reducción de información.
- C. Ninguna es correcta.

Aprendizaje Automático

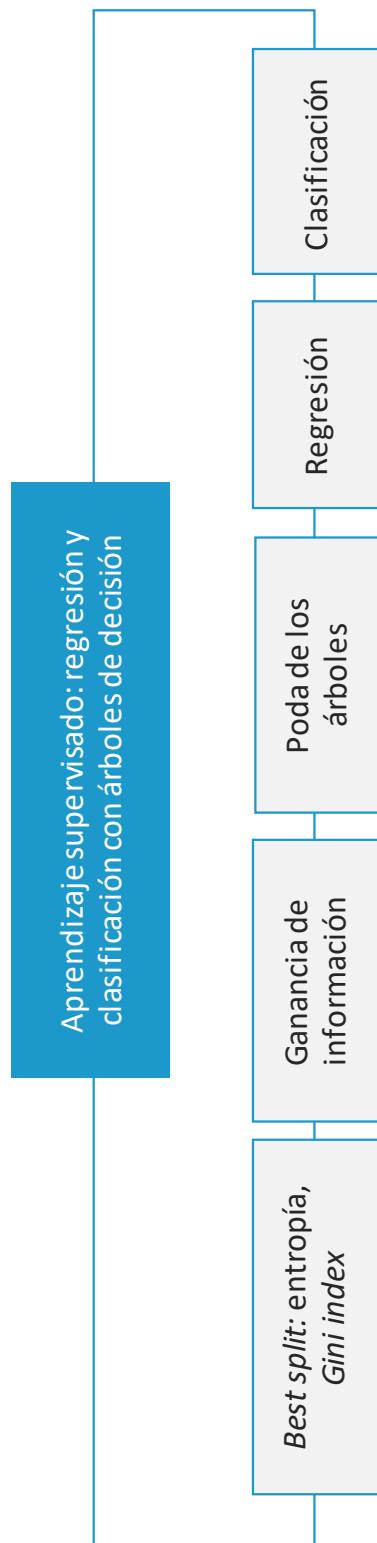
---

# Aprendizaje supervisado: regresión y clasificación con árboles de decisión

# Índice

|  |           |
|--|-----------|
| <b>Esquema</b>   | <b>3</b>  |
| <b>Ideas clave</b>   | <b>4</b>  |
| 5.1. ¿Cómo estudiar este tema?   | 4         |
| 5.2. Introducción a los árboles de decisión                            | 4         |
| 5.3. Best split: entropía, <i>Gini index</i> , ganancia de información | 7         |
| 5.4. Poda de los árboles   | 9         |
| 5.5. Árboles para clasificación  | 11        |
| 5.6. Árboles <i>vs.</i> modelos lineales                               | 13        |
| 5.7. Referencias bibliográficas  | 16        |
| <b>Lo + recomendado</b>  | <b>17</b> |
| <b>+ Información</b>   | <b>19</b> |
| <b>Test</b>  | <b>20</b> |

# Esquema



## 5.1. ¿Cómo estudiar este tema?

Estudia este tema a través de las **Ideas clave** disponibles a continuación.

**E**n este tema veremos los modelos de clasificación y regresión basado en los árboles de decisión.

- ▶ En primer lugar, veremos una introducción a los árboles de decisión.
- ▶ A continuación, se hará énfasis en cómo se decide el mejor corte, cubriendo para ello los conceptos de índice *Gini* y ganancia de información.
- ▶ Más adelante, se describe la necesidad de realizar la poda de los árboles.
- ▶ Posteriormente, se describen brevemente las diferencias con los árboles utilizados para clasificación.
- ▶ Finalmente se muestran las diferencias entre los árboles y un modelo de regresión lineal.

## 5.2. Introducción a los árboles de decisión

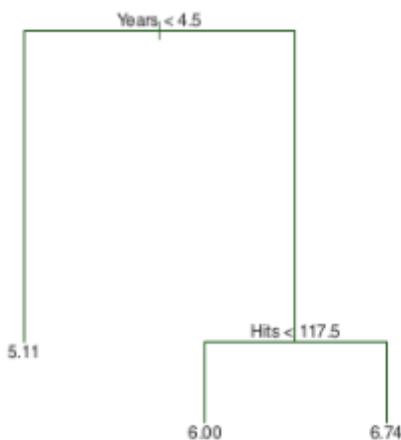
Los árboles de decisión son una de las **técnicas más populares dentro del campo del aprendizaje automático**. Se llevan utilizando durante **muchos años** en el área de la **minería de datos**. Se trata de métodos **simples** y fáciles de **interpretar**.

Estos modelos **dividen o segmentan** el espacio de las variables predictoras en una serie de **regiones**. Una vez creado el árbol de decisión es utilizado para predecir observaciones futuras. Para este propósito se utiliza la **moda** en el caso de que la variable a predecir sea categórica o bien la **media** en el caso de que sea numérica.

Como el conjunto de las reglas para separar las variables predictoras se pueden resumir en forma de árbol, a estos métodos se les conoce popularmente con el nombre de **árboles de decisión**.

Los árboles de decisión **dividen el espacio en rectángulos** que minimizan el error de la predicción. Debido a que es computacionalmente imposible considerar cualquier posible partición del espacio de entrada se utiliza un enfoque *top-down greedy* conocido como *recursive binary splitting*.

En la siguiente imagen se muestra un **árbol de decisión sencillo** para obtener el logaritmo del salario de los jugadores de béisbol. Este valor se obtiene por medio de las variables de años en la liga (*years*) y de golpes (*hits*) y utilizando unas reglas de decisión en base a los umbrales de estas variables. Así, por tanto, aquellos jugadores con más de 4,5 años de experiencia y más de 117,5 golpes estarían en 6,74, mientras que si tuvieran menos de 117,5 golpes estarían en 6,0.

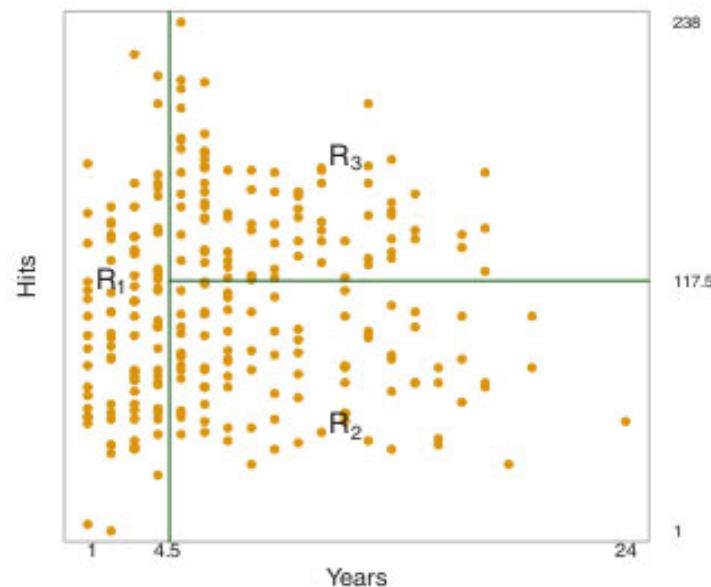


Gráfica 1. Ejemplo de un árbol de decisión sencillo para estimar el salario de los jugadores de béisbol en función de los años y de los golpes.

Fuente: James et.al., 2017.

El árbol de decisión anterior se obtiene por medio de la representación en un espacio de dos dimensiones de los puntos donde están el eje de años (*years*) y el de los golpes (*hits*).

A cada uno de esos puntos le corresponde un salario determinado y el objetivo es agrupar aquellos puntos que tienen un salario similar utilizando la información de las otras dos variables (golpes y años de experiencia).



Gráfica 2. Representación en dos dimensiones de la distribución de los golpes y años en la liga de los jugadores. Fuente: James et.al., 2017.

El árbol de decisión se obtiene **por medio de un algoritmo** que elige primero aquella variable que es más predictiva de la variable objetivo. A continuación, los ejemplos de entrenamiento se dividen en grupos con distintos valores para las clases de esta primera variable. El algoritmo continúa dividiendo los nodos con la elección de la mejor variable en cada iteración hasta que se alcance el criterio de parada. En cada iteración el algoritmo elige aquella variable que mejor predice la variable objetivo, por tanto, se trata de un algoritmo de tipo *greedy*. El criterio de parada puede venir dado por algunas de estas situaciones:

- ▶ Todos, o casi todos, los ejemplos del nodo son de la misma clase.
- ▶ No existen variables para distinguir entre los ejemplos.
- ▶ El árbol ha alcanzado un tamaño predefinido.

Existen numerosas implementaciones de los árboles de decisión disponibles en el mercado, no obstante, las más famosas son:

- ▶ C.5.0 desarrollado por Ross Quinlan.
- ▶ C 4.5.
- ▶ ID3 (*Iterative Dichotomiser 3*).
- ▶ J48, una alternativa basada en Java open source del C.4.5.

De todos ellos el algoritmo C.5.0 está considerado como el estándar en la industria.

Los árboles de decisión dividen o segmentan el espacio de las variables predictoras en una serie de regiones. En el caso de los árboles utilizados para modelos de regresión se utiliza la **media** para estimar los valores que se encuentran en una determinada región. En el caso de los modelos de clasificación se utiliza la **moda** de la clase.

## 5.3. Best split: entropía, Gini index, ganancia de información

Cada una de las divisiones del árbol da como resultado dos grupos. Es decir, los datos resultantes de la división por la variable elegida y con el punto de corte determinado están en uno de los dos grupos. Cuando los segmentos de una división contienen valores de una única clase se considera que es una **división pura**.

A la hora de identificar los criterios para realizar el *split* o corte existen varias métricas de pureza. Muchos algoritmos, por ejemplo, el C.5.0 utilizan el concepto de **entropía**. En este caso un valor de 0 indica que la muestra es completamente homogénea, mientras que un valor de 1 indica desorden completo.

La entropía se define con la siguiente fórmula matemática:

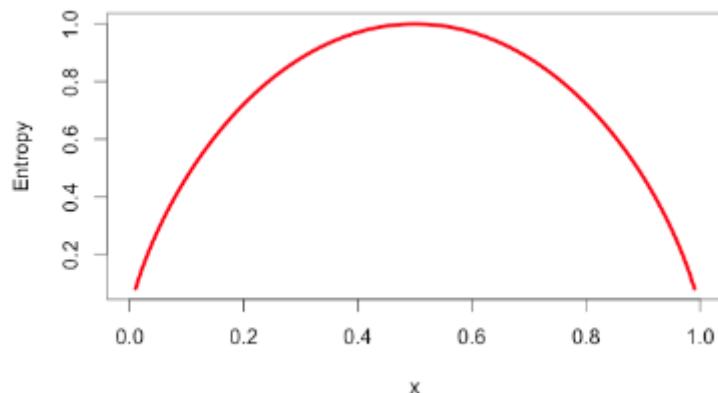
$$\text{Entropy}(S) = \sum_{i=1}^c -p_i \log_2(p_i)$$

Por ejemplo, si se ha realizado una partición utilizando una variable y umbral determinado y los datos de dos clases se dividen en un 60 % en un lado de la rama y en un 40 % en otro lado de la rama, tendríamos el siguiente valor de entropía:

```
> -0.60 * log2(0.60) - 0.40 * log2(0.40)
[1] 0.9709506
```

Si conocemos que la proporción de ejemplos de una clase es  $x$ , se puede examinar la entropía de todas las posibles divisiones de dos clases con el lenguaje de programación *R* utilizando la función *curve()*:

```
> curve(-x * log2(x) - (1 - x) * log2(1 - x), col = "red", xlab = "x", ylab = "Entropy", lwd = 4)
```



Gráfica 3. Ejemplo de los posibles valores de entropía de dos clases en función de las divisiones de cada una de las clases. Cuando la división entre las dos clases es cercana a 0,5 el valor de entropía es cercano a 1. Cuanto más cercana sea la división hacia la mayoría de una de las clases más cercano a cero es el valor de la entropía.

Fuente: James et.al., 2017.

Utilizando la medida de pureza el algoritmo tiene que decidir con que variable hacer el corte. Una opción es utilizar la entropía para calcular el cambio resultante de hacer el corte en esa variable.

A continuación, se calcula la ganancia de información (*information gain o IG*) que es la diferencia entre la entropía en el segmento antes de hacer el corte ( $S_1$ ) y la partición resultante de hacer el corte ( $S_2$ ), es decir:

$$\text{InfoGain}(F) = \text{Entropy}(S_1) - \text{Entropy}(S_2)$$

Después de un corte los datos se pueden dividir en más de una partición, por tanto, se considera la entropía a lo largo de las  $N$  particiones, ponderando la entropía de cada partición con el número de instancias de esa partición:

$$\text{Entropy}(S) = \sum_{i=1}^n W_i \text{Entropy}(P_i)$$

## 5.4. Poda de los árboles

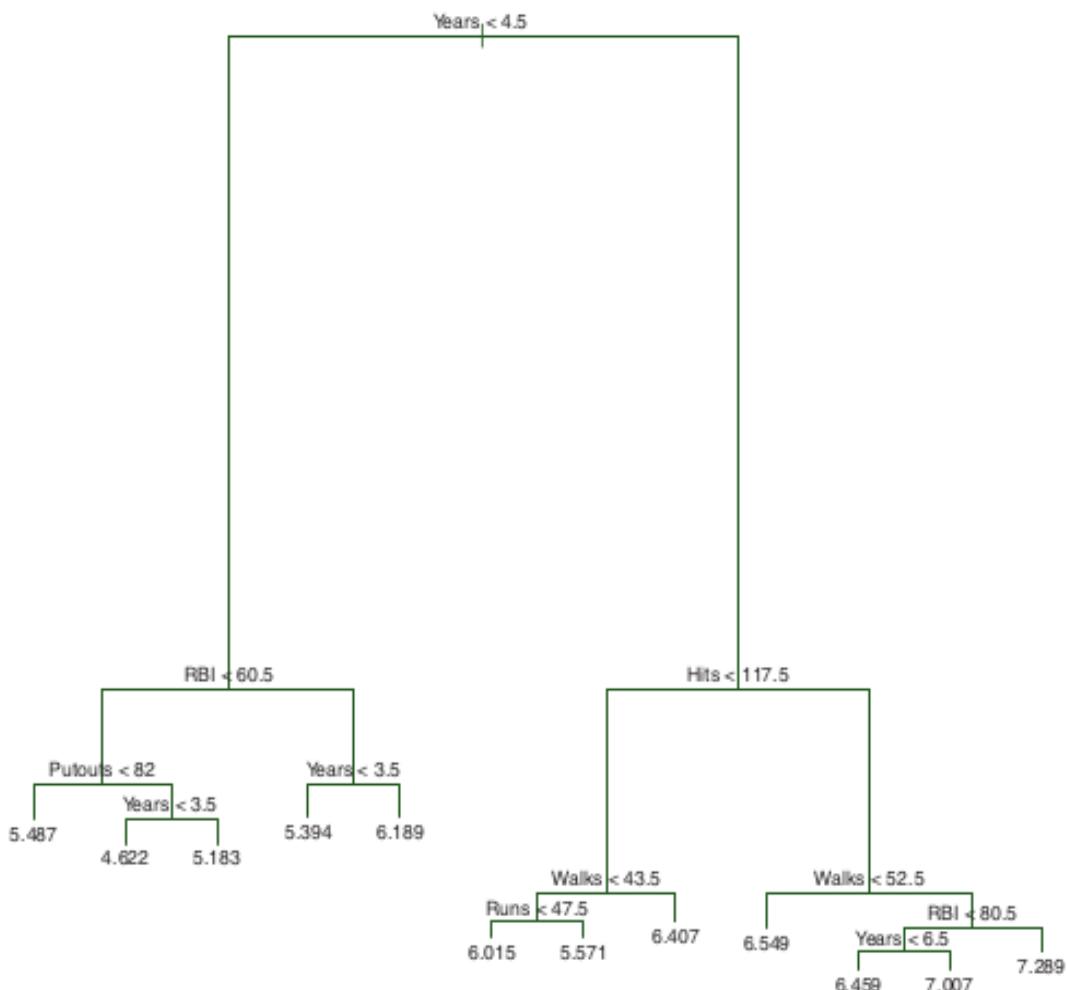
El proceso anterior de generación de cortes recursivos denominado *top-down greedy* puede generar buenas predicciones en el conjunto de entrenamiento, pero también sufre el **problema del sobre ajuste**.

Este motivo se debe a que el árbol resultante puede llegar a ser muy complejo y ajustarse muy bien a los datos de entrenamiento. Un árbol mucho más pequeño puede dar lugar a una menor **varianza** y mejor interpretación con el coste de un pequeño sesgo.

La estrategia para generar estos árboles más pequeños y con una menor varianza suele ser **generar un árbol muy grande** y después podarlo para obtener un sub-árbol (*post-prunning*).

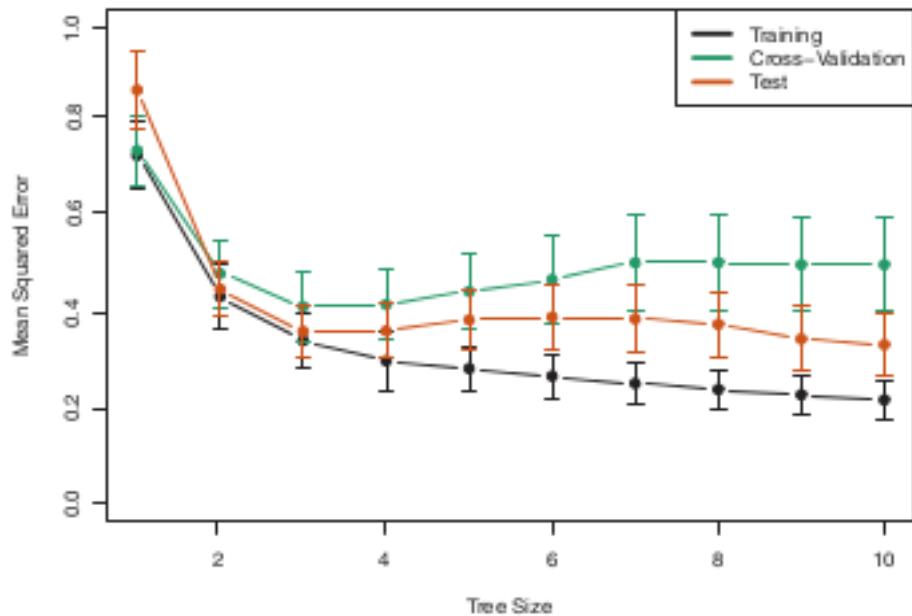
Ahora bien, la cuestión es, **¿cómo seleccionamos el sub-árbol?** Una buena solución es **utilizar aquel sub-árbol que proporcione un menor error de test en validación-cruzada (*cross validation*) o bien en el conjunto de validación.**

Por ejemplo, en la gráfica 4 se muestra la obtención de un árbol sin podar. En la gráfica 5 se muestra el error obtenido por este árbol en función de la profundidad para los conjuntos de entrenamiento (*train*), test y validación cruzada (*cross validation*). Se observa que el punto óptimo es utilizar tres niveles.



Gráfica 4. Ejemplo de un árbol sin podar con bastantes ramas.

Fuente: James et.al., 2017.



Gráfica 5. Ejemplo de la evolución del error del árbol de la figura 4 en función del número de niveles y para 3 conjuntos de datos. Fuente: James et.al., 2017.

## 5.5. Árboles para clasificación

Los árboles se pueden utilizar para realizar una predicción numérica en el caso de regresión o bien para realizar una clasificación. Los árboles de clasificación (*classification trees*) son similares a los *regression trees* pero predicen la respuesta de una variable categórica.

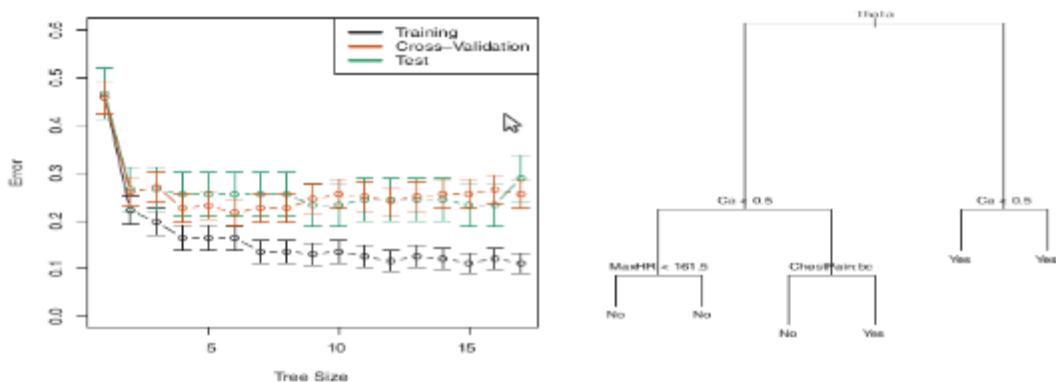
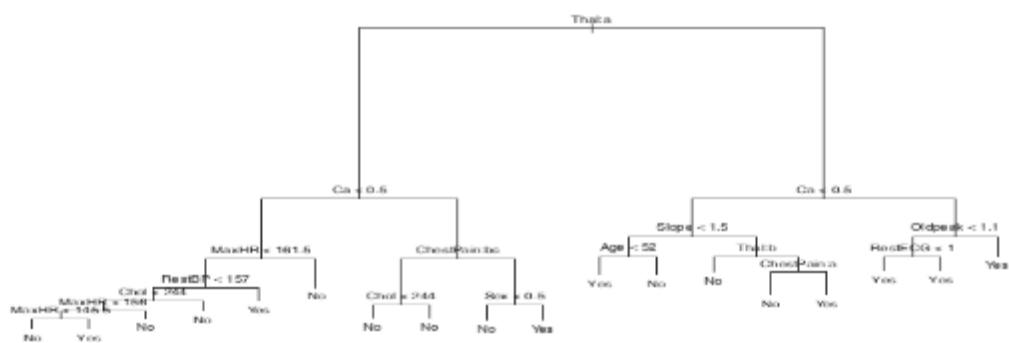
En *regression trees* la predicción corresponde a la media de las observaciones de ese nodo terminal. En *classification trees* se trata de la clase que tiene un mayor número de ocurrencias en ese nodo terminal.

En el caso de clasificación, el criterio que se suele utilizar para realizar los cortes o *splits* es el *Gini index*, que se define:

$$G = \sum_{K=1}^K \widehat{p_{mk}} (1 - \widehat{p_{mk}})$$

O bien la *entropía cruzada*, la cual se define como:

$$D = - \sum_{K=1}^K \widehat{p_{mk}} \log \widehat{p_{mk}}$$



Gráfica 6. Ejemplo de un árbol utilizado para realizado para clasificación. En la imagen superior se muestra el árbol completo. En la imagen inferior izquierda se muestra la evolución del error en función de la profundidad del árbol. En la imagen inferior derecha se observa el árbol podado resultante.

Fuente: James et.al., 2017.

## 5.6. Árboles vs. modelos lineales

Una regresión lineal asume un modelo que tiene la siguiente forma:

$$f(X) = \beta_0 + \sum_{j=1}^p X_j \beta_j$$

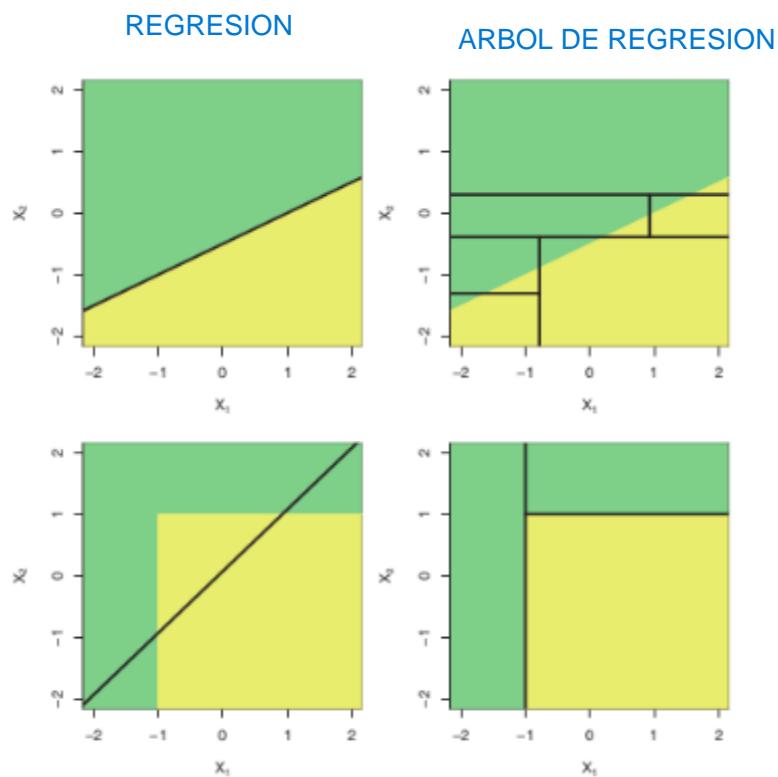
Mientras que un **árbol de regresión**, asume un modelo:

$$f(X) = \sum_{m=1}^M c_m \cdot 1(x \in R_m)$$

¿Qué modelo es mejor? Depende de lo que se esté **modelando**. Si las relaciones entre las variables de entrada y la variable respuesta se pueden aproximar por un modelo lineal, una regresión lineal funciona bien y supera a un árbol de regresión. Esto se debe a que un árbol de regresión no es capaz de modelar esta estructura.

Sin embargo, si la naturaleza del problema es no lineal y con relaciones complejas entre las variables, los árboles funcionan mejor.

Esta situación se refleja en la siguiente figura:



Gráfica 7. Ejemplo de ajustes a una región utilizando un modelo basado en árboles y uno basado en una regresión lineal. Si los puntos se distribuyen como en la imagen superior una regresión lineal (imagen izquierda superior) proporciona el mejor ajuste, sin embargo, un modelo en árbol (imagen izquierda derecha) se comporta mal. En el caso de una separación no lineal como ocurre en las imágenes inferiores esta situación se invierte.

Fuente: James et.al., 2017.

A continuación, se van a repasar los conceptos de los árboles de decisión y se van a discutir las ventajas e inconvenientes de su aplicación.



Ventajas e inconvenientes de los árboles de decisión.

---

Accede al vídeo a través del aula virtual

---

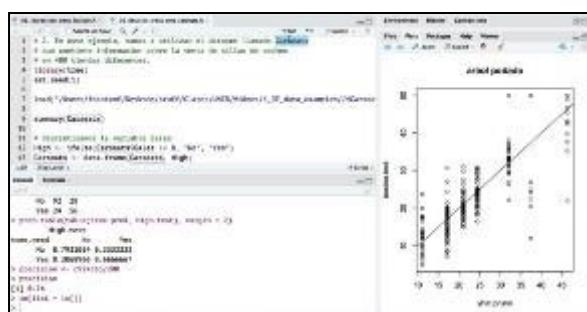
En este vídeo tutorial se analiza un ejemplo de árbol de regresión utilizando el paquete *tree*.

```
1 // Ejemplo de inserción con PreparedStatement
2 
3 // 1. Ejemplo de inserción como utilizando el comando insert
4 insert into empleados values('');
5 
6 // 2. Ejemplo de inserción usando PreparedStatement
7 PreparedStatement prep = null;
8 
9 prep = conn.prepareStatement("insert into empleados values(?)");
10 prep.setString(1, "Juan");
11 prep.setString(2, "Pérez");
12 prep.setString(3, "123456");
13 prep.setString(4, "2012-01-01");
14 prep.executeUpdate();
15 
16 // 3. Cargando el parámetro null que contiene el valor de las columnas
17 // de los NÚMEROS DE RÉGIMEN
18 prep.setString(1, "Juan");
19 prep.setString(2, "Pérez");
20 prep.setString(3, "123456");
21 prep.setString(4, "2012-01-01");
22 
23 consulta = "select * from empleados where Régimen=?";
24 resultado =
```

Tutorial Árboles de regresión.

Accede al vídeo a través del aula virtual

A continuación, en el siguiente vídeo tutorial se mostrarán los árboles de decisión aplicados a la clasificación.



Tutorial Árboles de clasificación.

Accede al vídeo a través del aula virtual

## 5.7. Referencias bibliográficas

James, G., Witten, D., Hastie, T. y Tibshirani, R. (2017). *An Introduction to Statistical Learning with Applications in R*. Springer.

# Lo + recomendado

## No dejes de leer

### Explicación visual funcionamiento de los árboles de decisión

R2D3. (s. f.). A visual introduction to machine learning. [mensaje en un blog].

Una explicación muy visual sobre el funcionamiento de los árboles de decisión con el objetivo de clasificar los alquileres en San Francisco frente a Nueva York.

Accede al artículo a través del aula virtual o desde la siguiente dirección web:

<http://www.r2d3.us/visual-intro-to-machine-learning-part-1/>

### Decisión Tree en Python utilizando Sklearn

Scikit learn. (s. f.). *Decision Trees*.

Explicación del uso de los árboles de decisión con de la librería Scikit learn en Python.

Accede al artículo a través del aula virtual o desde la siguiente dirección web:

<http://scikit-learn.org/stable/modules/tree.html>

## Ejemplo en Python de un árbol de decisión

McCarthy, J. (s. f.). Python for Data Science.

Creación de un árbol de decisión con los pasos necesarios del algoritmo desde cero y utilizando el lenguaje Python.

Accede al artículo a través del aula virtual o desde la siguiente dirección web:

[http://nbviewer.ipython.org/github/gumption/Python\\_for\\_Data\\_Science/blob/master/4\\_Python\\_Simple\\_Decision\\_Tree.ipynb](http://nbviewer.ipython.org/github/gumption/Python_for_Data_Science/blob/master/4_Python_Simple_Decision_Tree.ipynb)

**A comparative analysis of methods for pruning decision trees. IEEE Transactions on Pattern Analysis and Machine Intelligence**

Esposito, F., Malerba, D., Semeraro, G. y Kay, J. (1997). *A comparative analysis of methods for pruning decision trees*. IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol 19. pp. 476-491.

Artículo científico que compara los métodos utilizados para realizar la poda en los árboles de decisión y evitar el sobre ajuste

***An empirical comparison of selection measures for decision-tree induction***

Mingers, J. (1989). *An empirical comparison of selection measures for decision-tree induction*. Machine Learning. Vol 3, pp. 319-242.

Artículo científico que compara y evalúa las métricas utilizadas en la inducción en los árboles de decisión.

## Bibliografía

Quinlan, J. (1986). *Introduction to decision trees*. Nueva York: Springer.

1. Los árboles de decisión:

- A. Son una técnica nueva para solucionar problemas de clasificación.
- B. Son una técnica específica de los problemas de regresión.
- C. Se llevan utilizando durante muchos años en el área de la minería de datos.

2. Los árboles de decisión:

- A. Son métodos simples y difíciles de interpretar.
- B. Son métodos complejos y fáciles de interpretar.
- C. Son métodos simples y fáciles de interpretar.

3. Los árboles de decisión:

- A. Transforman las variables de entrada utilizando funciones *kernel*.
- B. Dividen el espacio de las variables en una serie de regiones.
- C. Ejecutan diversas operaciones modificando las variables de entrada.

4. Los árboles de decisión:

- A. Predicen la moda en el caso de un problema de regresión.
- B. Predicen la media en el caso de un problema de regresión.
- C. Predicen la moda en el caso de un problema de clasificación.
- D. Predicen la moda en el caso de un problema de regresión.

5. Un árbol de decisión se obtiene:

- A. Por medio de un algoritmo que elige primero una variable aleatoria.
- B. Por medio de un algoritmo que elige primero aquella variable más predictiva.
- C. Por medio de un algoritmo que elige primero aquella variable menos predictiva.

6. El criterio de parada de la construcción de un árbol de decisión viene dado por:

- A. Todos, o casi todos, los ejemplos del nodo son de la misma clase.
- B. No existen variables para distinguir entre los ejemplos.
- C. El árbol ha alcanzado un tamaño predefinido.



7. En el caso de la entropía:

- A. Un valor de 0 indica desorden completo.
- B. Un valor de 1 indica orden completo.
- C. Ninguna de las anteriores es correcta.

8. La poda de los árboles se realiza:

- A. Para reducir recursos computacionales.
- B. Para evitar el sobre-ajuste.
- C. Para generalizar las predicciones.

9. En los árboles de decisión para hacer la división:

- A. Se puede utilizar el criterio de *Gini index*.
- B. Se puede utilizar el criterio de entropía cruzada.
- C. Se puede utilizar el criterio de ganancia de información.

**10.** Cuáles de las siguientes afirmaciones son verdaderas:

- A. Un árbol de decisión y una regresión lineal pueden modelar problemas de similares.
- B. Un árbol de decisión modela bien problemas con dependencia no lineal.
- C. Una regresión lineal modela bien problemas con dependencia lineal.

Aprendizaje Automático

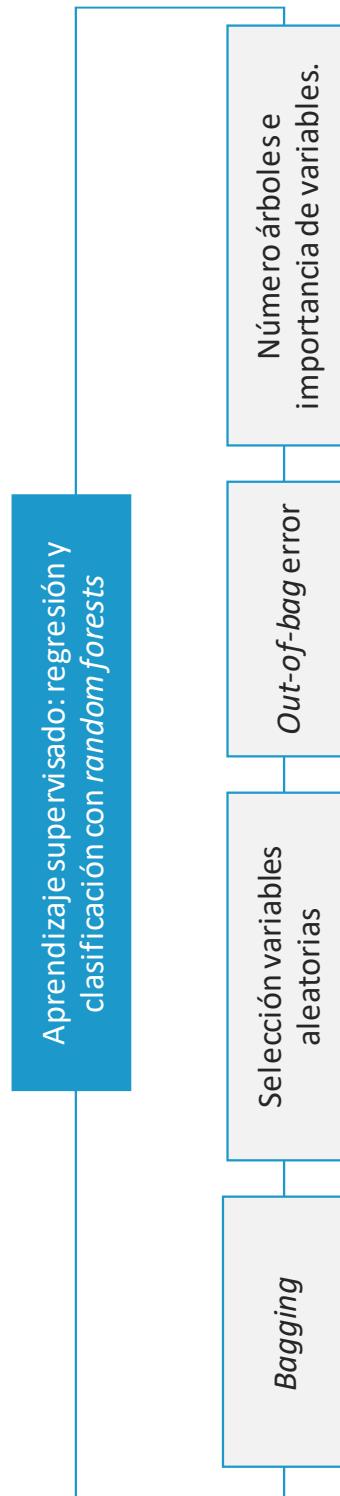
---

# Aprendizaje supervisado: regresión y clasificación con *random forests*

# Índice

|  |           |
|--|-----------|
| <b>Esquema</b>   | <b>3</b>  |
| <b>Ideas clave</b>   | <b>4</b>  |
| 6.1. ¿Cómo estudiar este tema?   | 4         |
| 6.2. Explotando la diversidad: <i>bagging</i> y selección de variables | 4         |
| 6.3. Interpretación de <i>out-of-bag</i> error                         | 7         |
| 6.4. Evolución del número de árboles e importancia de variables        | 8         |
| 6.5. Referencias bibliográficas  | 11        |
| <b>Lo + recomendado</b>  | <b>12</b> |
| <b>+ Información</b>   | <b>14</b> |
| <b>Test</b>  | <b>16</b> |

# Esquema



## 6.1. ¿Cómo estudiar este tema?

Estudia este tema a través de las **Ideas clave** disponibles a continuación.



*Random forests* es un **método basado en ensambles** de árboles de decisión que fue inventado por Leo Breiman y Adele Cutler en el año 2001 (Breiman, 2001).

En este tema se describe como los modelos de *random forests* explotan la diversidad y son capaces de generalizar y predecir mejor que un árbol de decisión. Esta diversidad se consigue por medio de **entrenar modelos con el método bagging y de realizar una selección de variables aleatoria**.

A continuación, se discute la interpretación del *out-of-bag error*. Posteriormente nos enfocamos en determinar el número de árboles y en evaluar la importancia de las variables.

## 6.2. Explotando la diversidad: *bagging* y selección de variables



Uno de los inconvenientes principales de los árboles de decisión es su **baja capacidad predictiva**. Este inconveniente se puede solventar por medio de utilizar un **ensamble** o **combinación de modelos de árboles de decisión**. El modelo *random forests* es en esencia un ensamble de árboles de decisión.

Los ensembles de árboles se pueden combinar utilizando los métodos de *bagging* o *boosting*. Los modelos de *random forests* se basan en combinar los árboles por medio de los métodos de *bagging*.

En el caso de problemas de clasificación en cada observación de test se almacena la clase predicha por cada uno de los B árboles y la clase final se obtiene por medio del voto de la mayoría. Es decir, la **predicción global** es la clase que más veces ocurre a lo largo de las B predicciones. Por otro lado, en el caso de los problemas de regresión la **predicción global** es la media de las predicciones de cada uno de los árboles.

El modelo de *random forests* combina los principios de *bagging* con selección de variables aleatorias para añadir diversidad a los árboles de decisión. Una vez es generado el ensemble de árboles (*forest*) el modelo utiliza el mecanismo de votación o la media para generar las predicciones.

Se trata de un modelo que **combina versatilidad y potencia en un enfoque**. A la hora de construir cada uno de los árboles se utiliza una porción pequeña y aleatoria de las variables de entrada disponibles, por lo general, este número se define como  $\sqrt{p}$  siendo  $p$  el número de variables de entrada.

Al ser un modelo que genera cada árbol con un subconjunto de los registros de entrada y una selección aleatoria de las variables, **puede trabajar con conjuntos** de datos bastante **grandes y no se encuentra afectado por los problemas de curse of dimensionality**. Además, debido a que los árboles son modelos con mucho ruido y muy inestables se pueden beneficiar de forma notoria de ser promediados.

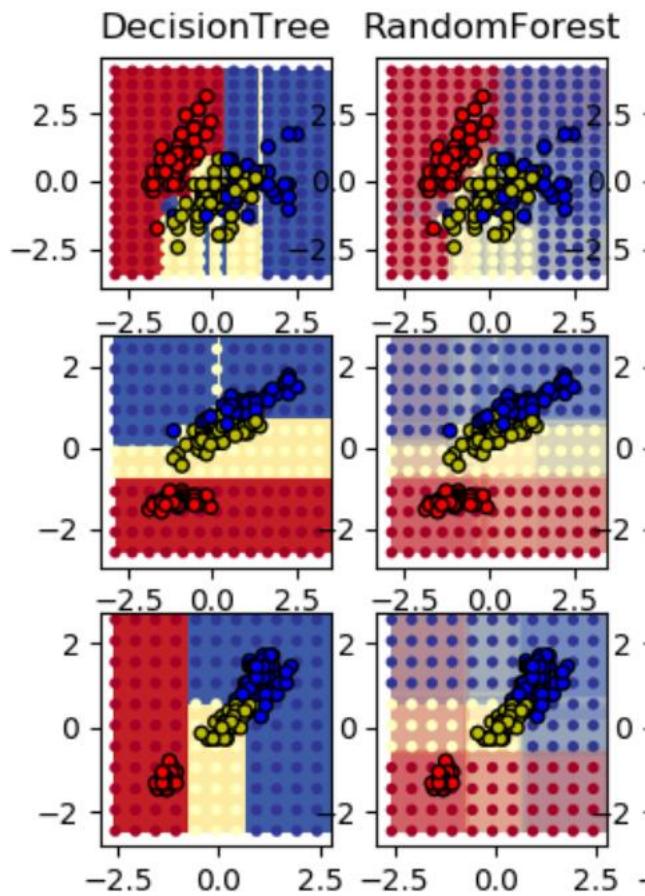
En cada uno de los cortes de cada uno de los árboles se elige una serie de variables candidatas de entre todas las posibles. El valor por defecto,  $\sqrt{p}$  variables aleatorias de entre todas las posibles, suele dar buenos resultados.

Este modelo se basa en la **utilización de un gran número de árboles**. La razón de utilizar un gran número de árboles es para que cada variable, de entre todas las posibles, tenga la oportunidad de aparecer en varios modelos.

La ventaja principal de los modelos de *random forests* frente a un modelo *bagged de árboles* es debido a que las variables de cada *split* de los árboles se obtienen por medio de **seleccionar un subconjunto de todas las variables de forma aleatoria**. De esta forma, **se consigue decorrelar los árboles generados**. Además, se reduce la varianza porque el resultado se obtiene a partir de calcular el promedio de los árboles y se controla el sobreajuste.

Por tanto, cada árbol de decisión se construye utilizando *bootstrapped training samples* y en cada *split* se utiliza una variable a partir de una selección aleatoria de  $m$  predictores del total  $p$ . Finalmente, los distintos árboles se combinan utilizando el voto de la mayoría para clasificación y la media para regresión.

En los *random forests* debido a la selección de variables aleatorias sobre el conjunto de entrada completo, **el sesgo de los árboles se suele incrementar** (con respecto al sesgo un único árbol de decisión); no obstante, **debido a promediar los árboles la varianza se reduce** y, por lo general, en una proporción mayor que el incremento del sesgo dando lugar a un mejor modelo por lo general. En muchos problemas el rendimiento de los *random forests* es similar a los modelos *boosting* a pesar de ser más sencillos de entrenar. Como consecuencia de este hecho, los *random forests* se han convertido en una técnica muy popular.



Gráfica 1. Regiones detectadas por un árbol de decisión y un modelo de *random forests* sobre el *dataset* de iris. Recuperado de [http://scikit-learn.org/stable/auto\\_examples/ensemble/plot\\_forest\\_iris.html](http://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_iris.html)

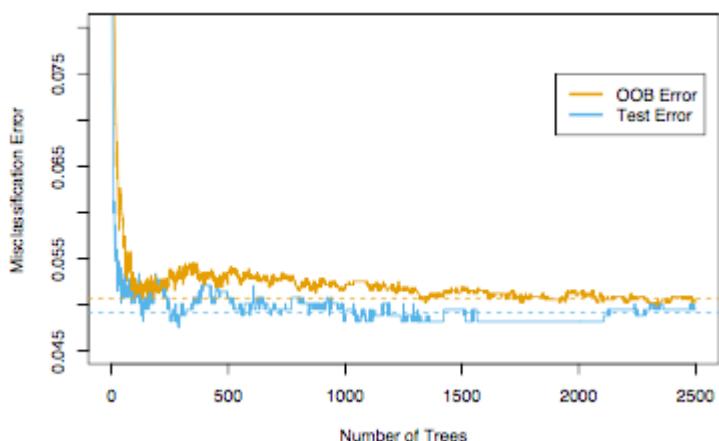
### 6.3. Interpretación de *out-of-bag* error

**U**na característica importante de los modelos *random forests* es el error *out-of-bag*. El *out-of-bag* error es una forma sencilla de estimar el error de test en un modelo *bagged*. Se puede demostrar que de media cada árbol construido con un modelo bagged utiliza  $2/3$  de las observaciones del conjunto de entrenamiento. El  $1/3$  restante de las observaciones de entrenamiento no se utilizan para generar el árbol *bagged* y son llamadas *out-of-bag*.

Para cada observación se puede obtener la respuesta de los *bagged trees*, donde esta observación era *out-of-bag*. Esto proporciona alrededor de  $B/3$  predicciones para cada una de las observaciones.

Si el valor de  $B$  (número de árboles) es grande esta estimación es equivalente a un modelo de tipo *leave-one-out* de validación cruzada. Este modelo de *leave-one-out* es la forma más rigurosa y extrema de evaluar un modelo de aprendizaje automático.

Un modelo *random forests* se puede entrenar de forma secuencial, donde no es necesario utilizar validación cruzada, pues las muestras *out-of-bag* proporcionan una estimación similar. Para ello, es necesario observar la evolución del *out-of-bag* error y una vez que este error se haya estabilizado el algoritmo puede parar el entrenamiento.

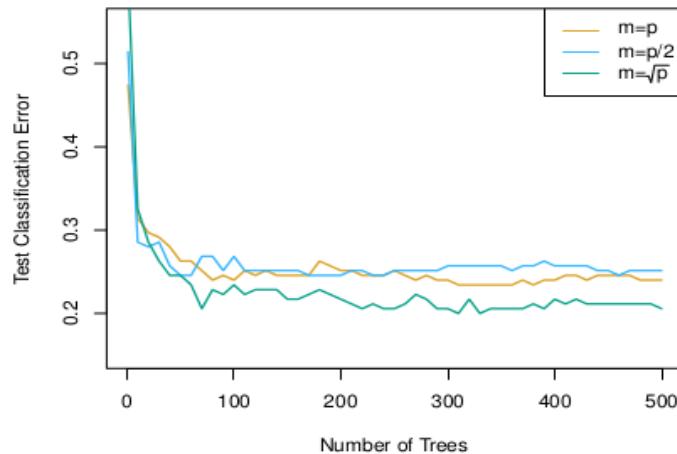


Gráfica 2. Evolución del *out-of-bag* error. La gráfica muestra la evolución de 2500 árboles, pero se observa que 200 árboles serían suficientes. Fuente: Hastie et. al., 2001.

## 6.4. Evolución del número de árboles e importancia de variables

**L**os modelos basados en *random forests* principalmente tienen **dos parámetros para optimizar**: el número de árboles y el número de variables que se evalúan en cada tirada. Existen otros parámetros como la profundidad de los árboles, pero en la práctica no presentan muchos cambios.

En la siguiente gráfica se observa la evolución del error en el conjunto de test en función del número de árboles y del número de variables que se prueban en cada *split*. Se observa que el valor de  $\sqrt{p}$  proporciona un error menor que los otros valores.



Gráfica 3. Tres diferentes resultados de un *random forests* para un problema de clasificación de 15 clases y que tiene 500 predictores. Fuente: James et. al., 2017.

En principio se puede pensar que cuanto más grande sea el número de árboles mejor. Sin embargo, por lo general existe un **punto óptimo** donde a pesar de añadir más árboles el error no se reduce de forma significativa.

Una de las ventajas de los *random forests* es que pueden generar buenos resultados sin necesidad de muchos ajustes manuales. Además, permiten construir de forma sencilla gráficos de importancia de variables.

En cada decisión de corte de cada árbol la mejora en el criterio de corte es la medida de importancia que se atribuye al **corte de esa variable** y se agrega para todos los árboles del *forest* para cada variable.

*Random forest* también utiliza las muestras fuera del *bag* (OOB) para construir una medida de importancia de variables que mide la capacidad predictiva de cada una de las variables.

En el siguiente vídeo se van a discutir las ventajas y desventajas de los *random forest*.



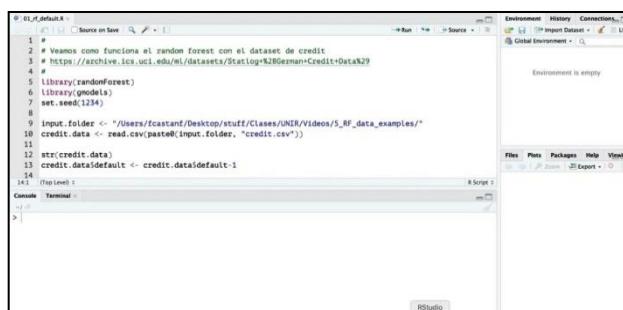
Ventajas e inconvenientes de los *random forest*.

---

#### Accede al vídeo a través del aula virtual

---

A continuación, en el vídeo tutorial se verá cómo se ejecuta y configura un modelo de *random forest* utilizando el lenguaje de programación R y el entorno de desarrollo RStudio.



Tutorial *Random forest*.

---

#### Accede al vídeo a través del aula virtual

---

## 6.5. Referencias bibliográficas

Breiman, L. (2001). Random Forests. *Machine Learning*. 45(1), 5-32.

Hastie, T., Tibshirani, R. y Friedman, J. (2001). *The elements of Statistical Learning*. Springer.

James, G., Witten, D., Hastie, T. y Tibshirani, R. (2017). *An Introduction to Statistical Learning with Applications in R*. Springer.

# Lo + recomendado

## No dejes de leer

### ***Random Forests in Python***

Yhat. (5 de junio de 2013). Random Forests in Phyton [Mensaje en un blog].

Blog con información y ejemplos de *random forests* en Python.

---

Accede a la entrada a través del aula virtual o desde la siguiente dirección web:

<http://blog.yhat.com/posts/random-forests-in-python.html>

---

### **Random Forests**

Breiman, L. y Cutler, A. (s. f.). Random Forests.

Información sobre *random forests* de la Universidad de Berkeley.

---

Accede al artículo a través del aula virtual o desde la siguiente dirección web:

[https://www.stat.berkeley.edu/~breiman/RandomForests/cc\\_home.htm](https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm)

---

# No dejes de ver

## Machine learning – Random Forests

Clase sobre *random forests* que contiene ejemplos de detección de objetos y de Kinect.

### Outline of the lecture

This lecture discusses classification trees and how to incorporate them into an ensemble (random forest). It discusses:

- Random trees
- Random forests
- Object detection
- Kinect

---

Accede al vídeo a través del aula virtual o desde la siguiente dirección web:

<https://www.youtube.com/watch?v=3kYujfDgmNk>

---

## A fondo

### **Random Forests in Python**

Yhat. (5 de junio de 2013). Random Forests in Python. [Mensaje en un blog].

Blog con información y ejemplos de *random forests* en Python.

---

Accede al post a través del aula virtual o desde la siguiente dirección web:

<http://blog.yhat.com/posts/random-forests-in-python.html>

---

## Webgrafía

### **Random Forests**

Información sobre *random forests* de la Universidad de Berkeley.

**Random Forests  
Leo Breiman and Adele Cutler**

---

Accede a la página web a través del aula virtual o desde la siguiente dirección web:

[https://www.stat.berkeley.edu/~breiman/RandomForests/cc\\_home.htm](https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm)

---

## Bibliografía

Breiman, L. (2001). Random forests. *Machine Learning*, 45, 5-32.

Lei Zhang, F. H. (2019). The use of classification and regression algorithms using the random forests method with presence-only data to model species' distribution. *MethodsX*, 2281-2292.

1. El método *random forests*:

- A. Se basa en los árboles de decisión y fue inventado en los años 90.
- B. Se basa en el método *bagging* y en los árboles de decisión.
- C. Fue inventado en el año 2001 por Leo Breiman.

2. El método *random forests*:

- A. Combina las técnicas de *bagging* con la selección de variables aleatorias para incluir diversidad.
- B. Es un método de ensemble basado en árboles de decisión.
- C. Se basa en los principios de los números aleatorios para realizar las predicciones.

3. Los modelos de *random forests*:

- A. En los problemas de clasificación predicen utilizando la moda de la clase más probable.
- B. En los problemas de regresión predicen utilizando la media de las predicciones de cada árbol.
- C. Tanto en los problemas de clasificación como regresión utilizan la mediana de las predicciones.

4. El motivo de utilizar un gran número de árboles en *random forests*:

- A. Cuantos más árboles mejor funciona la aleatoriedad.
- B. Las variables tienen más probabilidades de aparecer en los diferentes cortes.
- C. Ninguna de las anteriores es correcta.

**5.** En los *random forests* respecto de los árboles de decisión:

- A. La varianza de los árboles se suele incrementar.
- B. La varianza de los árboles es el mismo.
- C. La varianza de los árboles suele decrecer.**

**6.** En los *random forests* respecto de los árboles de decisión

- A. El sesgo de los árboles se suele incrementar.**
- B. El sesgo de los árboles es el mismo.
- C. El sesgo de los árboles suele decrecer.

**7.** El *out-of-bag* error:

- A. Consiste en aquellos errores que se quedan fuera de un umbral.
- B. Es una forma sencilla de estimar el error en test en un modelo *bagged*.**
- C. Está formado por los errores en cada una de las iteraciones.

**8.** Un modelo de *random forests*:

- A. Hay que entrenarlo siempre con validación cruzada.
- B. Se puede entrenar**
- C. Ninguna de las anteriores es correcta.

**9.** En cuanto al número de árboles de un modelo de *random forests*:

- A. Cuantos más árboles tenga mejor.
- B. Existe un punto óptimo en lo referente al número de árboles.**
- C. La capacidad predictiva siempre es la misma a partir de 100 árboles.

**10.** Los *random forests* tienen como parámetros:

- A. Principalmente el número de árboles y el número de variables aleatorias elegidas en cada corte.**
- B. Bastantes parámetros que son muy sensibles y afectan a su precisión.
- C. Ninguna de las anteriores es correcta.

Aprendizaje Automático

---

# Combinación de clasificadores: *bootstrapping, bagging y boosting*

# Índice

|                                      |           |
|--------------------------------------|-----------|
| <b>Esquema</b>                       | <b>3</b>  |
| <b>Ideas clave</b>                   | <b>4</b>  |
| 7.1. ¿Cómo estudiar este tema?       | 4         |
| 7.2. Introducción                    | 4         |
| 7.3. Técnica de <i>bootstrapping</i> | 5         |
| 7.4. Método <i>bagging</i>           | 6         |
| 7.5. Método de <i>boosting</i>       | 8         |
| 7.6. Referencias bibliográficas      | 12        |
| <b>Lo + recomendado</b>              | <b>13</b> |
| <b>Test</b>                          | <b>16</b> |

# Esquema



## 7.1. ¿Cómo estudiar este tema?

Estudia este tema a través de las **Ideas clave** disponibles a continuación.

**E**n este tema se van a estudiar los **métodos de ensemble**. Estos métodos **combinan las predicciones de varios estimadores base**, que han sido construidos con un algoritmo de aprendizaje automático, para mejorar la generalización/robustez de un único modelo.

- ▶ En primer lugar, a partir del capítulo de introducción se va a introducir la técnica estadística de *bootstrapping*.
- ▶ A continuación, se describe el método *bagging*, el cual se apoya en la técnica de *bootstrapping* para ensamblar modelos.
- ▶ Finalmente, veremos el método de *boosting*, uno de los métodos más populares para llevar a cabo la combinación de modelos débiles para construir un modelo más robusto y con mayor precisión.

## 7.2. Introducción

**P**or lo general los **métodos de ensemble** o combinación de clasificadores se pueden dividir en dos grandes familias:

1. *Averaging methods*: los cuales se basan en **construir varios estimadores de forma independiente y luego promediar las predicciones**. En media, **el estimador combinado se suele comportar mejor que cualquier estimador individual porque la varianza se reduce**. Los métodos de *bagging* funcionan de esta forma.

2. ***Boosting methods***: los estimadores base se construyen de forma secuencial con el objetivo de reducir el sesgo del estimador combinado. La motivación es combinar varios modelos débiles para producir un modelo combinado potente.

### 7.3. Técnica de *bootstrapping*

*Bootstrapping* es una **técnica estadística** que consiste en cualquier test o métrica sobre muestreo aleatorio con remplazamiento. *Bootstrapping* **permite asignar medidas de precisión** definidas en términos de sesgo, varianza, intervalos de confianza o cualquier otra métrica sobre estimaciones muestrales. Esta técnica permite la estimación de la distribución de la muestra de cualquier estadístico utilizando métodos de muestreo. Generalmente, se suele clasificar como una clase de método de muestreo.

*Bootstrapping* es la **práctica de estimar las propiedades de un estimador** (así como su varianza) **por medio de medir las propiedades al muestrear una distribución aproximada**. Una elección habitual para aproximar la distribución es utilizar la función de distribución empírica de los datos observados. En el caso de que las observaciones se puedan asumir que se generan a partir de una población independiente e idénticamente distribuidas, pueden ser obtenidas por medio de muestreos con remplazamiento del conjunto de datos observados.

La idea sobre la que se basa la técnica de *bootstrapping* es que la inferencia sobre una población a partir de una muestra de datos puede ser **modelada por medio de re-muestrear los datos de la muestra** y llevar a cabo inferencias sobre un remuestreo de la muestra. Como la población es desconocida el error verdadero en una muestra estadística sobre el valor de la población es desconocido. En el remuestreo en *bootstrapping*, la población es en realidad la muestra, la cual es conocida. Por tanto, la calidad de la inferencia de la muestra verdadera a partir de los datos muestreados es medible.

Como ejemplo, supongamos que nos interesa conocer la **media de la altura de la población mundial**. Debido a que es **complejo medir la altura de todas las personas en el mundo, en su lugar podemos muestrear una parte de ellas y medir la altura en ellas**. Supongamos que el tamaño de esta muestra es  $N$ , entonces tenemos la altura de  $N$  personas. Con esta muestra solo podemos tener una estimación de la altura media de la población. Para obtener una mejor imagen de la población necesitamos obtener algún tipo de variabilidad sobre los datos obtenidos. El método de *bootstrap* más sencillo para obtener esta variabilidad consiste en muestrear con remplazamiento una muestra de tamaño  $N$ . Por ejemplo, si muestreamos sobre [1,2,3,4,5] podríamos obtener [2,5,3,3,1]. Cuando el tamaño  $N$  es suficientemente grande para todos los efectos prácticos existe una probabilidad casi cero de que sea una muestra idéntica a la muestra inicial. Si se repite este proceso miles de veces y para cada muestra se obtiene la media, se obtiene un histograma de medias obtenido con *bootstrap*.

El método *bagging*, descrito a continuación, **consiste en un meta-algoritmo para ponderar los resultados de múltiples muestras de bootstrapping.**

## 7.4. Método *bagging*

**E**l método de *bagging* es una clase de algoritmos de ensemble o combinación de clasificadores. El término *bagging* proviene de la contracción de *bootstrap aggregation*. Se trata de un procedimiento general que permite reducir la varianza de un método de *machine learning*. Es un método para **combinar varias instancias de estimadores de caja negra** que se han construido sobre muestras aleatorias del conjunto de entrenamiento original y que agregan las predicciones individuales para obtener una predicción única.

Dado un conjunto de  $n$  observaciones independientes  $Z_1 \dots Z_n$ , cada una con una varianza  $\sigma$ , la varianza de la media  $Z$  de las observaciones es  $\sigma/n$ .

Es decir, la media de un conjunto de observaciones reduce la varianza. Sin embargo, como lo habitual es no tener muchas observaciones se suele hacer *bootstrap* tomando muestras repetidas del *data set* de entrenamiento.

Para ello, se entrena  $B$  conjuntos de entrenamiento distintos y se obtiene la media de las predicciones:

$$\hat{f}_{bag}(x) = \frac{1}{B} \sum_{b=1}^B \hat{f}^{*b}(x)$$

Estos métodos se utilizan como una forma para reducir la varianza de un estimador base (ejemplo: árboles de decisión) por medio de introducir aleatoriedad en el procedimiento de construcción y a continuación construir el ensemble.

En muchos casos, los métodos de *bagging* son un método sencillo de mejorar, un único modelo sin la necesidad de tener que modificar el algoritmo de entrenamiento base. Como se trata de un método para reducir el *overfitting*, los métodos de *bagging* funcionan mejor cuando se utilizan con modelos complejos (ejemplo: árboles de decisión profundos), en contraste con los métodos de *boosting* que funcionan mejor con modelos simples (ejemplo: shallow decisión trees).

Existen diferentes variantes de los métodos de *bagging*, pero dependen principalmente de la forma en que obtienen las muestras del conjunto de entrenamiento:

- ▶ Cuando se obtienen muestras aleatorias del conjunto de datos como subconjuntos aleatorios de las muestras, el algoritmo se conoce como *pasting* (Breiman, 1999).
- ▶ Cuando se obtienen los conjuntos para entrenar los algoritmos por medio de muestreo con remplazamiento del conjunto de entrenamiento, el método se conoce como *bagging* (Breiman, 1998).
- ▶ Cuando se obtienen subconjuntos aleatorios como subconjuntos del espacio de las variables, el método se conoce como *random subspaces* (Kam, 1998).

- ▶ Por último, cuando los estimadores base se construyen con subconjuntos de muestras y variables, el método se conoce como *random patches* (Louppe. y Geurts, 2012).

### ***Out-of-bag (OOB) error***

Existe una forma sencilla de estimar el error de test en un modelo *bagged*, **por medio del conjunto de datos que se queda fuera de la muestra**. Este conjunto de datos fuera de la muestra, por lo general, suele ser 1/3 del conjunto de datos total. Por tanto, para este 1/3 del conjunto total se puede predecir la respuesta que se hubiera obtenido. Cuando el valor  $B$  (número de conjuntos de entrenamiento distintos) es grande, la estimación del *out-of-bag* error es equivalente al *leave-one-out cross validation*.

## **7.5. Método de *boosting***

Al igual que *bagging*, se trata de un **método de combinación de modelos** que se puede aplicar a los modelos de regresión y de clasificación.

En *bagging* se crean múltiples copias del conjunto de entrenamiento original utilizando el muestreo *bootstrap* y entrenando un modelo en cada copia, para posteriormente combinar todos los modelos en uno solo. En *bagging* cada modelo se construye en un *bootstrap data set* independiente de los otros. Sin embargo, en el caso de *boosting* los modelos se generan de forma secuencial, es decir, cada modelo utiliza información de los modelos anteriores.

*Boosting* es la técnica seguida por los modelos de *generalized boosted models* (GBMs). En estos modelos, en lugar de entrenar un gran árbol de decisión sobre los datos, lo cual es complicado y puede dar lugar a *overfitting*, se utiliza el enfoque *boosting* para aprender poco a poco. Cada uno de los árboles puede ser bastante pequeño, con pocos nodos terminales, pero todos ellos se combinan.

Por tanto, el método *boosting* puede verse como un **meta-algoritmo de ensemble** para reducir sesgo y varianza el cual se aplica sobre algoritmos de *machine learning* supervisados con el objetivo de convertir modelos simples en modelos más precisos. Este método se basa en una pregunta formulada por Kearns y Valiant (año 1989): «¿Puede un conjunto de modelos débiles crear un modelo fuerte?».

En este contexto un **modelo débil** se define como un **clasificador que está poco correlado con la clase real**. Se trata de ejemplos ligeramente mejor que el acierto aleatorio. Por otro lado, un modelo fuerte se define como un clasificador que está bien correlado con la clase verdadera de la clasificación.

La respuesta a la pregunta de Kearns y Valiant ha tenido una gran repercusión en el mundo de *machine learning* dando lugar al desarrollo de los modelos de *boosting*. Cuando se introdujo esta hipótesis de problemas *boosting* dio lugar a que los algoritmos que consiguieran mejorar por medio de la combinación de modelos débiles fueran llamados *boosting*.

A pesar de que los algoritmos *boosting* no tienen ninguna restricción algorítmica, la mayoría de estos algoritmos consisten en ir aprendiendo modelos débiles iterativamente con respecto a una distribución. Cada vez que se añade un nuevo modelo débil, los datos son re-calibrados: los ejemplos que estaban correctamente clasificados pierden peso y los ejemplos incorrectamente clasificados ganan peso.

## Tipos de algoritmos *boosting*

Existen diversos algoritmos de tipo *boosting*. El algoritmo original fue propuesto por Yoav Freund y Robert Schapire en 1997, el cual era no adaptativo y no explotaba de forma completa la ventaja de los modelos débiles. Posteriormente, Freund y Schapire desarrollaron el algoritmo *AdaBoost*, un algoritmo de *boosting* adaptativo que ganó el prestigioso premio Gödel.

La principal variación entre los algoritmos de *boosting* es el **método de ponderar los datos de entrenamiento y las hipótesis**. *AdaBoost* es muy popular e históricamente el primero que fue capaz de adaptarse a los modelos débiles. Sin embargo, hay más algoritmos de *boosting* como: el *LPBoost*, *BrownBoost*, *TotalBoost*, *LogitBoost* y *xgboost*.

## Gradient Boosting (Simple Version)

(Why is it called “gradient”?)  
(Answer next slides.)

(For Regression Only)

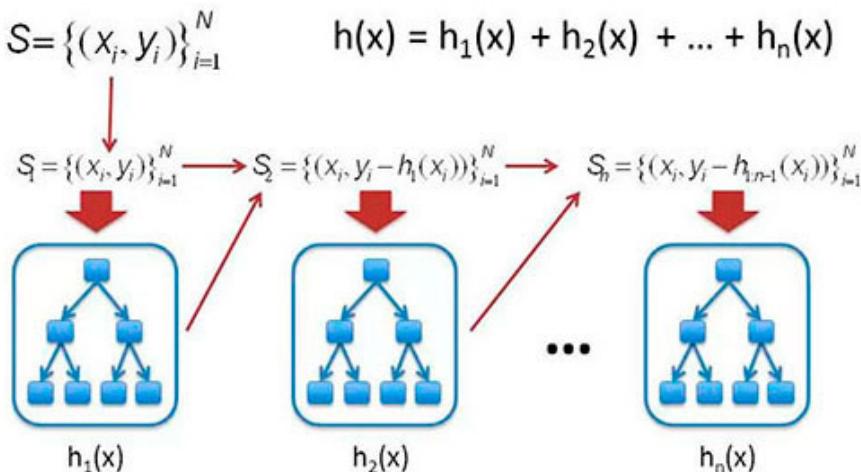


Figura 1. Ejemplo ilustrativo del funcionamiento de gradiente *boosting* para problemas de regresión.  
Recuperado de <http://statweb.stanford.edu/~jhf/ftp/trebst.pdf>

1. Set  $\hat{f}(x) = 0$  and  $r_i = y_i$  for all  $i$  in the training set.
2. For  $b = 1, 2, \dots, B$ , repeat:
  - 2.1 Fit a tree  $\hat{f}^b$  with  $d$  splits ( $d + 1$  terminal nodes) to the training data  $(X, r)$ .
  - 2.2 Update  $\hat{f}$  by adding in a shrunken version of the new tree:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x).$$

- 2.3** Update the residuals,



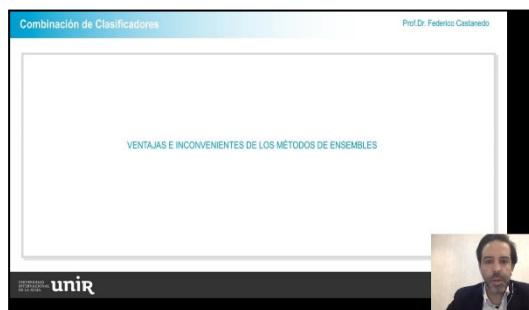
$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i).$$

3. Output the boosted model,

$$\hat{f}(x) = \sum_{b=1}^B \lambda \hat{f}^b(x).$$

Figura 3. Algoritmo *boosting*. Fuente: James et. al., 2017.

En el vídeo que se muestra a continuación se va a discutir el porqué de la necesidad de los métodos de ensembles y las ventajas e inconvenientes de los métodos de estos.



Ventajas e inconvenientes de los métodos de ensembles.

---

Accede al vídeo a través del aula virtual

---

## 7.6. Referencias bibliográficas

Breiman, L. (1998). Bagging predictors. *Machine Learning*, 24(2), 123-140.

Breiman, L. (1999). Pasting Small votes for classification in large databases and on-line. *Machine Learning*, 36(1), 85-103.

Freund, Y. y Schapire, R. (1997). A decisión-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1), 119-139.

James, G., Witten, D., Hastie, T. y Tibshirani, R. (2017). *An Introduction to Statistical Learning with Applications in R*. Springer.

Kam, T. (1998). The random subspace method for constructing decisión forests. *Pattern Analysis and Machine Intelligence*, 20(8), 832-844.

Louppe, G. y Geurts, P. (2012). Ensembles on Random Patches. *Machine Learning and Knwoledge Discovery in Databases*, 346-361.

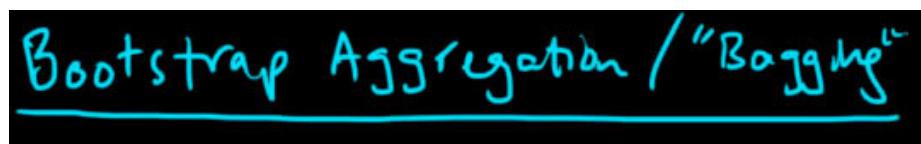
Quinlan, R. (2006). Bagging, boosting and C4.5. University of Sidney. Recuperado de <http://www.cs.ecu.edu/~dingq/CSCI6905/readings/BaggingBoosting.pdf>

# Lo + recomendado

## No dejes de ver

### Bootstrap aggregation (Bagging)

Vídeo con una explicación matemática de la técnica de *bagging*.



---

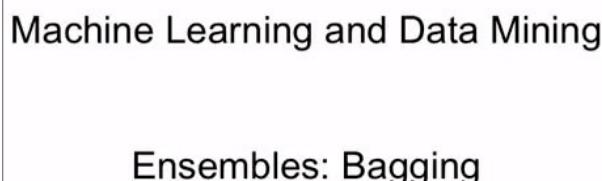
Accede al vídeo a través del aula virtual o desde la siguiente dirección web:

<https://www.youtube.com/watch?v=5Lu1eTiX7qM>

---

### Ensembles (2): Bagging

Vídeo de Alexander Ihler sobre el funcionamiento del método *bagging* y algunos ejemplos en Matlab.



---

Accede al vídeo a través del aula virtual o desde la siguiente dirección web:

<https://www.youtube.com/watch?v=Rm6s6gmLTdg>

---

## **Ensembles (4): AdaBoost**

Vídeo de Alexander Ihler sobre el método *boosting* y en concreto *AdaBoost*.

Machine Learning and Data Mining

Ensembles: Boosting

---

Accede al vídeo a través del aula virtual o desde la siguiente dirección web:

<https://www.youtube.com/watch?v=ix6IvwbVpwo>

---

## **Extending Machine Learning Algorithms – AdaBoost Classifier**

Vídeo sobre el método AdBoost y el Gradient Boosting Classifier con ejemplos en R y Python.

Boosting and Ensemble  
of Ensembles

---

Accede al vídeo a través del aula virtual o desde la siguiente dirección web:

<https://www.youtube.com/watch?v=BoGNyWW9-mE>

---

## **Learning: Boosting**

Clase del MIT del profesor Patrick Winston sobre *boosting*.

## **Learning: Boosting**

---

Accede al vídeo a través del aula virtual o desde la siguiente dirección web:

<https://www.youtube.com/watch?v=UHBmv7qCey4>

---

1. Los métodos de ensembles se pueden dividir en:

- A. *Average* y *random methods*.
- B. *Average* y *boosting methods*.
- C. *Average* y *gradient methods*.

2. La técnica de *bootstrapping*:

- A. Es una técnica estadística que consiste en obtener una métrica sobre una muestra aleatoria con reemplazamiento.
- B. Permite la estimación de la distribución de la muestra de cualquier estadístico.
- C. Ninguna de las anteriores es correcta.

3. En la técnica de *bootstrapping*:

- A. La inferencia de la población se lleva a cabo re-muestreando los datos de la muestra.
- B. Como el valor de la muestra es conocido, la calidad de la inferencia de la muestra verdadera a partir de los datos muestreados es medible.
- C. Ninguna de las anteriores es correcta.

4. El método de *bagging*:

- A. Es una técnica de ensembles que se puede aplicar únicamente a tareas de clasificación.
- B. Es una técnica de ensembles que se puede aplicar únicamente a tareas de regresión.
- C. Es una técnica de ensembles que se puede utilizar en modelos de clasificación o regresión.

**5.** El método *bagging*:

- A. Se puede utilizar sin necesidad de modificar los clasificadores base.
- B. Necesita que se modifiquen los algoritmos de los clasificadores base.
- C.** Agrega las predicciones de varios clasificadores para llevar a cabo una predicción única.

**6.** El método *bagging*:

- A. Se utiliza siempre junto con el método *boosting*.
- B. Mejora los modelos por medio de combinarse con el método *boosting*.
- C.** Se trata de un método para reducir el *overfitting*.

**7.** En el método *boosting*:

- A.** Los modelos se generan de forma secuencial e incremental.
- B. Se muestran los datos de entrada para obtener diversos clasificadores.
- C.** Se trata de un meta-algoritmo para reducir sesgo y varianza.

**8.** El método de *boosting*:

- A. Combina los clasificadores más potentes para mejorar la precisión.
- B.** Utiliza una serie de clasificadores débiles.
- C.** Es incremental donde nuevos modelos mejoran o se mantienen respecto de iteraciones previas.

**9.** Cuáles de los siguientes son algoritmos de *boosting*:

- A.** *TotalBoost*.
- B. *OptimalBoosting*.
- C.** *AdaBoost*.

**10.** El algoritmo original de *boosting*:

- A.** Fue propuesto por Freund y Schapire en 1997.
- B. Fue propuesto por Freund y Shapire en 1977.
- C. Ninguna de las anteriores es correcta.

Aprendizaje Automático

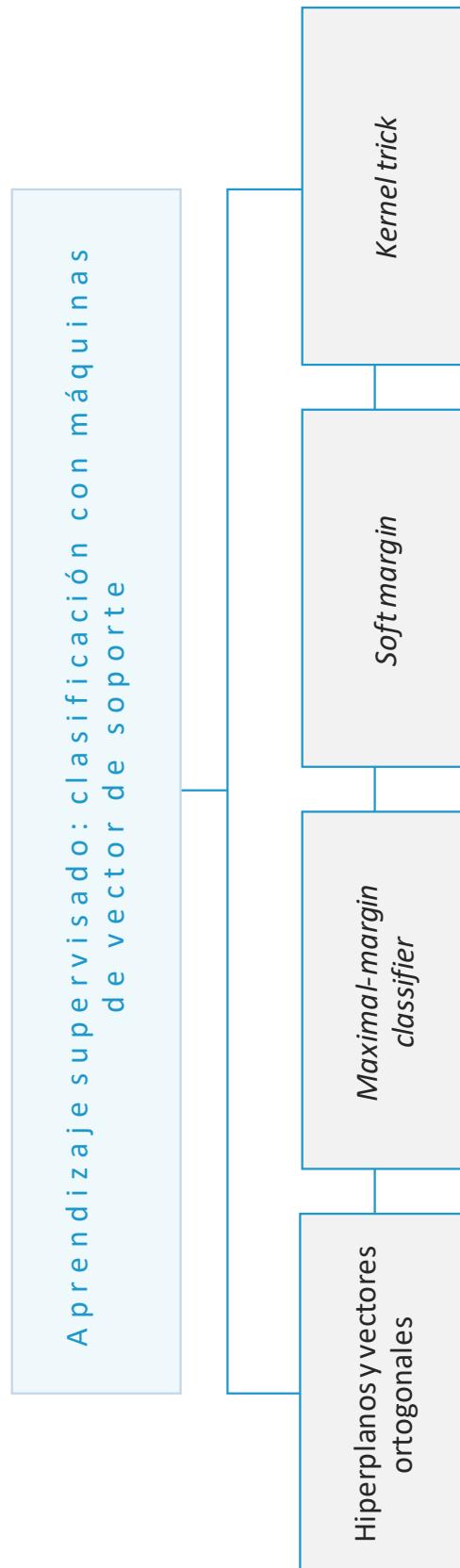
---

# Aprendizaje supervisado: clasificación con máquinas vector de soporte

# Índice

|  |           |
|--|-----------|
| <b>Esquema</b>   | <b>3</b>  |
| <b>Ideas clave</b>   | <b>4</b>  |
| 8.1. ¿Cómo estudiar este tema?                                     | 4         |
| 8.2. Introducción a las máquinas de vector de soporte: hiperplanos | 4         |
| 8.3. Separando por hiperplanos                                     | 7         |
| 8.4. <i>Maximal-margin classifier</i>                              | 8         |
| 8.5. <i>Soft margin</i>  | 11        |
| 8.6. <i>Kernel trick</i>   | 15        |
| 8.7. Referencias bibliográficas                                    | 17        |
| <b>Lo + recomendado</b>  | <b>18</b> |
| <b>+ Información</b>   | <b>20</b> |
| <b>Test</b>  | <b>21</b> |

# Esquema



## 8.1. ¿Cómo estudiar este tema?

Estudia este tema a través de las **Ideas clave** disponibles a continuación.

**E**n este tema vamos a estudiar las **máquinas de vector de soporte** y cómo se pueden aplicar en **problemas de clasificación dentro del aprendizaje supervisado**.

Las **máquinas de vector de soporte** han sido el primer modelo que utilizaba un **enfoque no probabilístico para realizar la estimación de datos no observados**.

- ▶ En primer lugar, vamos a ver los conceptos geométricos necesarios para entender el funcionamiento y la base que hay por detrás de las máquinas de vector de soporte.
- ▶ En segundo lugar, veremos los problemas existentes al separar por hiperplanos.
- ▶ A continuación, veremos cómo solucionar el problema de separar por hiperplanos.
- ▶ Finalmente, las técnicas conocidas como *kernel trick*.

## 8.2. Introducción a las **máquinas de vector de soporte: hiperplanos**

**L**as máquinas vectores de soporte conocidas como **support vector machines** fueron inventadas por el científico de origen ruso Vladimir Vapnik y su equipo en el año 1963.

La gran contribución de Vapnik al campo del aprendizaje automático ha consistido precisamente en proponer uno de los primeros modelos de estimación/predicción que no está basado en ningún modelo probabilístico, lo cual es un gran cambio de mentalidad. Puesto que todos los modelos que se usaban hasta ese momento estaban basados en la teoría estadística y de probabilidad.

Sin embargo, las máquinas de vector de soporte (SVM) se pueden considerar el primer modelo que utiliza un enfoque completamente distinto y que está basado en un modelado geométrico y que se resuelve mediante un problema de optimización con restricciones.

El objetivo de las redes de vectores de soporte es buscar un plano que separe las clases en *feature space*. Por *feature space* se entiende un nuevo espacio de dimensiones diferente (por lo general con un mayor número de dimensiones) al espacio original.

Debido a que este objetivo de buscar un plano que separe las clases por lo general no es posible obtener, en las SVM se proponen las siguientes modificaciones:

1. Relajar la definición de «separar».
2. Mejorar y enriquecer el *feature space* para que la separación sea posible.

Antes de entrar en detalle en el funcionamiento de las máquinas de vector de soporte es necesario definir los conceptos de hiperplanos y vectores ortogonales.

## Hiperplanos

En un espacio de dos dimensiones el hiperplano es una recta. Pero, un hiperplano se puede definir para un espacio de  $p$  dimensiones. La ecuación general del hiperplano para un espacio de  $p$  dimensiones es:

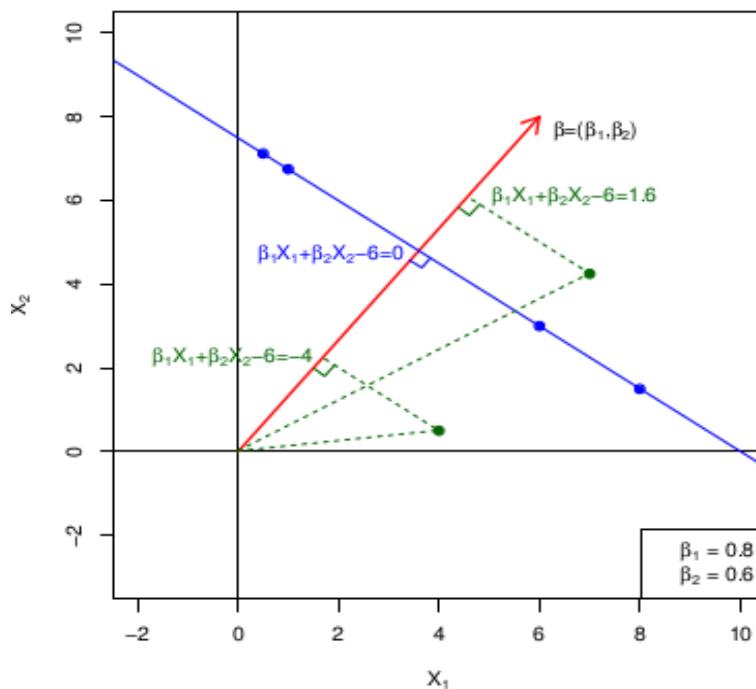
$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = 0$$

donde el vector:

$$\beta = (\beta_1, \beta_2, \dots, \beta_p)$$

Se llama vector normal, y  $\beta$  es un vector unitario en el cual la suma de los cuadrados es 1. Este vector apunta a una dirección ortogonal a la superficie del hiperplano (vector de color rojo de la figura 1.).

Si proyectamos cada uno de los puntos sobre el vector normal, los puntos que caen sobre el hiperplano tienen valor 0 al proyectarse. Los puntos por encima del hiperplano tienen un valor positivo, los puntos por debajo del hiperplano un valor negativo y además estos valores son mayores cuanto más lejanos están del hiperplano. Es decir, el valor que se obtiene al proyectar los puntos sobre el vector normal es proporcional a la distancia de los puntos al hiperplano. Esta característica geométrica hace posible el uso de las máquinas de vector de soporte para buscar los patrones necesarios.



Gráfica 1. Representación de un hiperplano, un vector normal y la distancia de los puntos proyectados en el vector normal. Recuperado de:

<https://lagunita.stanford.edu/courses/HumanitiesSciences/StatLearning/Winter2016/about>

La cuestión es que el hiperplano busca separar dos conjuntos de puntos en dos regiones distintas. Por ejemplo, en un problema de clasificación el hiperplano puede separar aquellos puntos que corresponden a personas con un tumor determinado de aquellas personas sanas.

Sin embargo, para un conjunto de puntos determinados existen múltiples hiperplanos posibles que nos separan los puntos en dos regiones.

El objetivo de las máquinas de vector de soporte es buscar un hiperplano que separe las clases en *feature space*. Por *feature space* se entiende un nuevo espacio de dimensiones diferente al espacio original.

### 8.3. Separando por hiperplanos

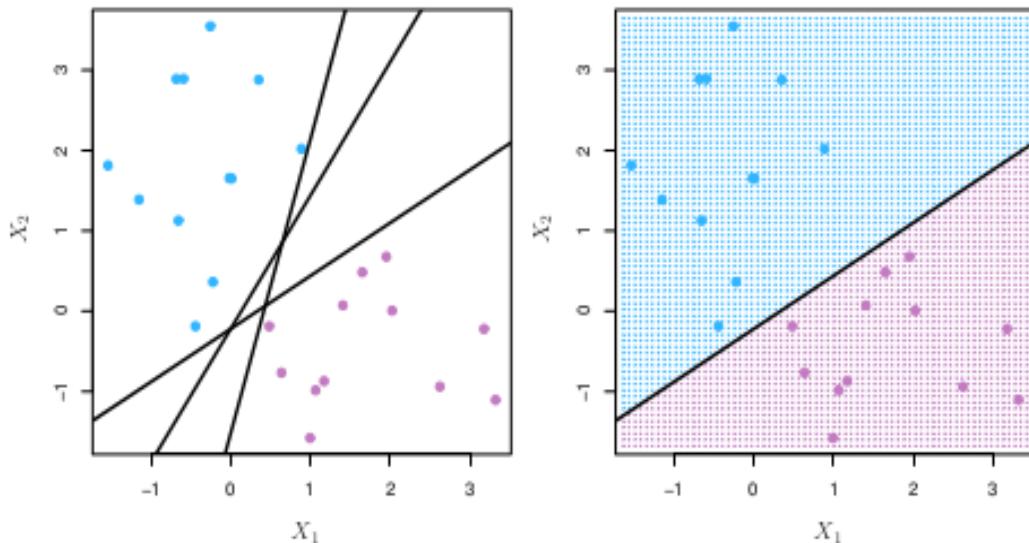
Dado una serie de puntos en un espacio geométrico que se desea clasificar, se puede establecer que aquellos puntos que al proyectar sobre el vector normal a un hiperplano determinado sean  $> 0$ , correspondan a una clase y aquellos que sean  $< 0$ , a otra clase.

Es decir, de forma matemática:

$$\text{Si } f(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p \text{ entonces}$$

$f(X) > 0$  establece los puntos en un lado del hiperplano  
 $f(X) < 0$  en el otro

Sin embargo, lo habitual es encontrarse en la situación en la cual existen múltiples hiperplanos posibles, y por tanto es necesario decidir ¿cuál de ellos establecer?



Gráfica 2. Ejemplo de tres hiperplanos posibles para separar dos conjuntos de clase (izquierda). En el caso de elegir un hiperplano los puntos se clasifican en función de cada una de las regiones sombreadas de rojo y azul (derecha). Fuente: James et.al., 2013.

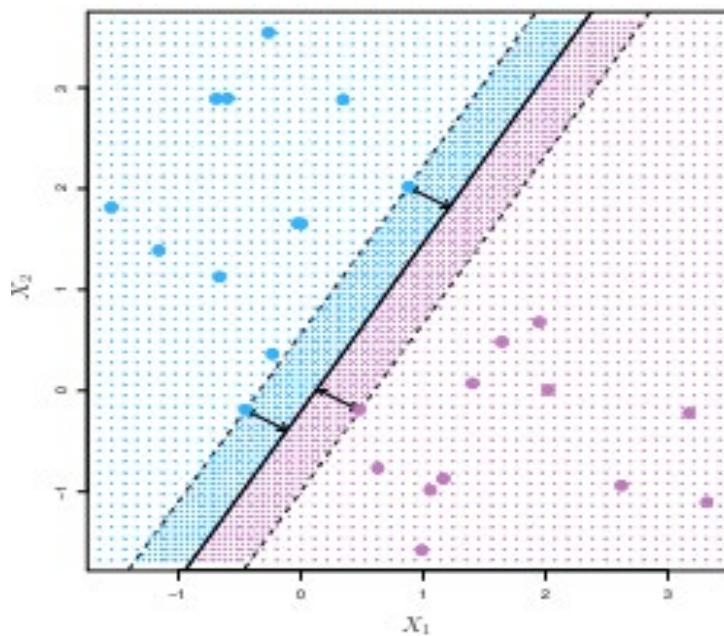
## 8.4. Maximal-margin classifier

**E**n el caso de un problema de clasificación binaria, de todos los hiperplanos posibles es necesario buscar aquel que nos proporciona la mayor diferencia entre las dos clases, lo cual se traduce en la mayor distancia entre los puntos que pertenecen a una clase y a otra. La hipótesis que hay detrás de esto, es porque suponemos que este hiperplano será el que tendrá una mayor distancia en el conjunto de test y en las predicciones futuras.

Esta situación se puede modelar como un problema de optimización con restricciones donde es necesario maximizar el margen y, matemáticamente, se define:

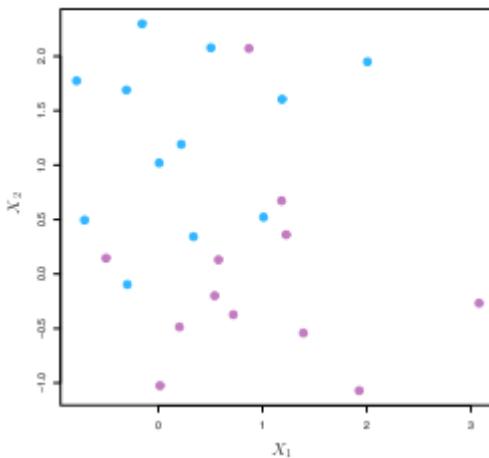
$$\begin{aligned}
 & \text{maximize } M \\
 & \beta_0, \beta_1, \dots, \beta_p \\
 & \text{subject to } \sum_{j=1}^p \beta_j^2 = 1 \\
 & y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) \geq M \\
 & \text{for all } i = 1, \dots, N
 \end{aligned}$$

En la siguiente imagen se muestra de forma gráfica el concepto donde la línea continua resaltada es el *maximal-margin classifier*, y se muestran dos bandas con líneas discontinuas que contienen la distancia del hiperplano a los primeros puntos de cada una de las clases, el objetivo es maximizar la distancia de estas dos bandas.



Gráfica 3. Ejemplo del hiperplano con una separación óptima entre dos conjuntos de puntos de clases diferentes. Fuente: James et.al., 2013.

El principal problema con esta separación óptima del hiperplano es que **los datos habitualmente no son linealmente separables con una recta** en el caso de un espacio de dos dimensiones, como se observa en la siguiente gráfica, donde es imposible separar los puntos de una clase de los de otra.



Gráfica 4. Ejemplo de un espacio de dos dimensiones donde los puntos no son linealmente separables.

Fuente: James et.al., 2013.

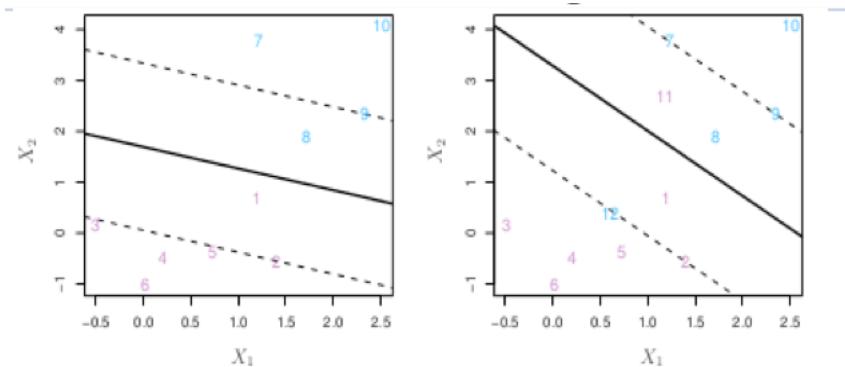
Esta situación produce una solución pobre para el clasificador *maximal-margin*. Por tanto, el clasificador vector de soporte maximiza un *soft margin*.

## 8.5. Soft margin

**L**a solución al problema anterior en el cual los puntos no son linealmente separables es maximizar un *soft margin*. Matemáticamente el problema se define de la siguiente forma:

$$\begin{aligned} & \text{maximize } M \\ & \beta_0, \beta_1, \dots, \beta_p, \epsilon_1, \dots, \epsilon_n \\ & \text{subject to } \sum_{j=1}^p \beta_j^2 = 1 \\ & y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) \geq M(1 - \epsilon_i) \\ & \epsilon_i \geq 0, \sum_{i=1}^n \epsilon_i \leq C \end{aligned}$$

De forma gráfica, en la siguiente figura se observa:

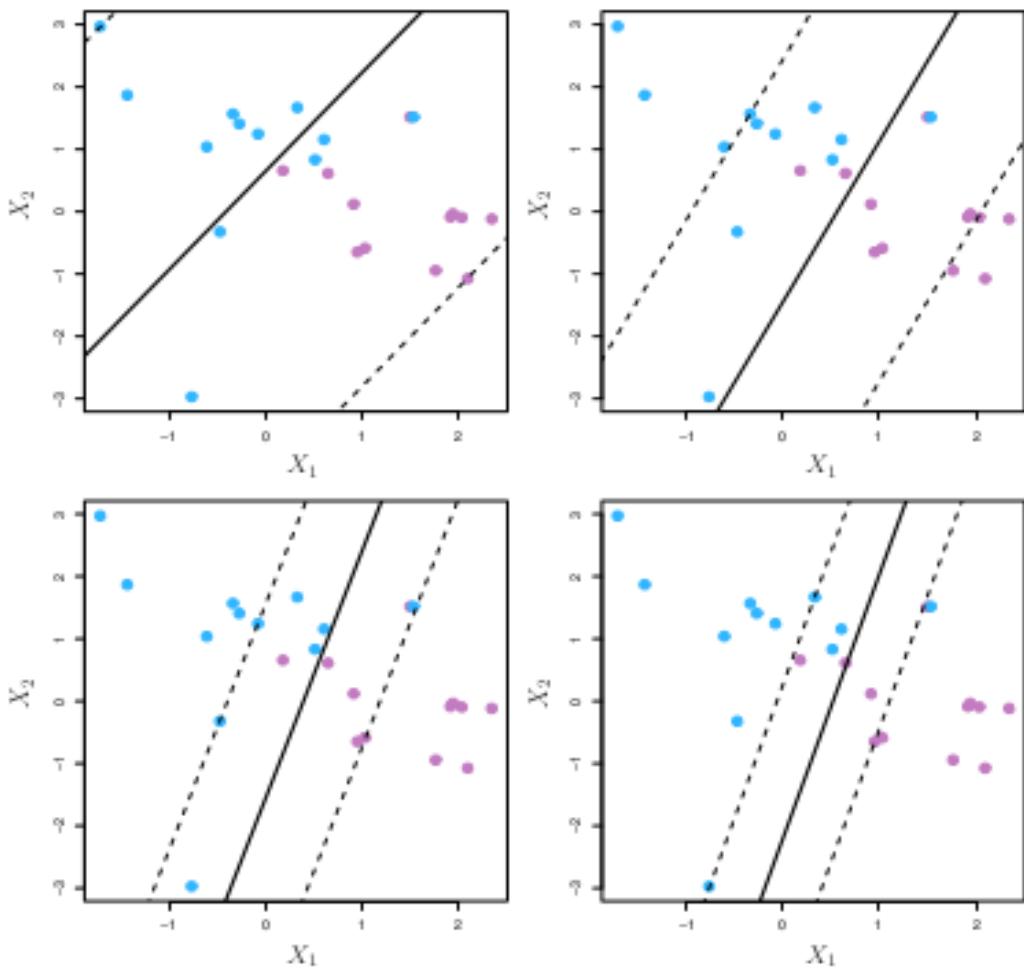


Gráfica 5. Ajuste de un clasificador de vector de soporte a pocos datos. El hiperplano se muestra con una línea sólida y los márgenes con una punteada. La imagen de la derecha muestra el efecto cuando aparecen dos nuevas observaciones (11 y 12) las cuales están en el lado incorrecto del hiperplano y de los márgenes.

Fuente: James et.al., 2013.

Modificando el parámetro  $C$  de la ecuación anterior, el cual se conoce como función de coste se puede hacer el margen más grande o pequeño. En concreto un valor de  $C$  más grande hace el margen más pequeño y a la inversa un valor  $C$  más pequeño hace el margen más grande.

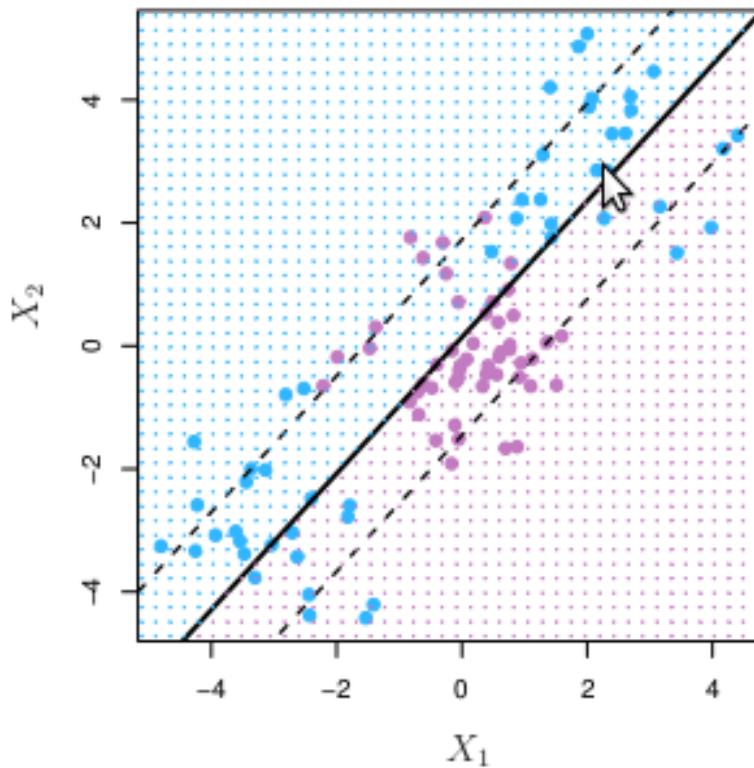
En la siguiente figura se muestra un hiperplano y los márgenes correspondientes en función de diferente valor de  $C$ .



Gráfica 6. Diferentes fronteras de decisión en los márgenes de una *soft-margin* en función de diferentes valores de  $C$  y utilizando los mismos puntos y el mismo hiperplano. El margen más grande es el de la gráfica superior izquierda y el más pequeño el de la inferior derecha. Fuente: James et.al., 2013.

### Problema con las fronteras lineales

Muchas veces las fronteras lineales siguen sin funcionar, independientemente del valor de coste  $C$  que se utilice. Por ejemplo, en la siguiente gráfica se muestra un problema donde la separación necesaria entre los puntos de una clase y la otra es no lineal.



Gráfica 7. Ejemplo donde no es posible una separación lineal entre dos clases. Fuente: James et.al., 2013.

### Expansión de variables

Para solucionar este problema donde no es posible realizar una separación entre dos clases, una opción es **incrementar el espacio de las variables por medio de transformaciones**, es decir pasar de un espacio **p-dimensional** a un espacio de D dimensiones  $D > p$ . Una vez realizada esta transformación, se debería ajustar un clasificador de vector de soporte en este nuevo espacio. El **objetivo es obtener fronteras de decisión no lineales sobre el espacio original**.

Por ejemplo, si tenemos datos de entrada en dos dimensiones  $(X_1, X_2)$  es posible utilizar el siguiente espacio de 6 dimensiones:  $(X_1, X_2, X_1^2, X_2^2, X_1X_2)$  siendo por tanto la frontera de decisión:

$$\beta_0 + \beta_1X_1 + \beta_2X_2 + \beta_3X_1^2 + \beta_4X_2^2 + \beta_5X_1X_2 = 0$$

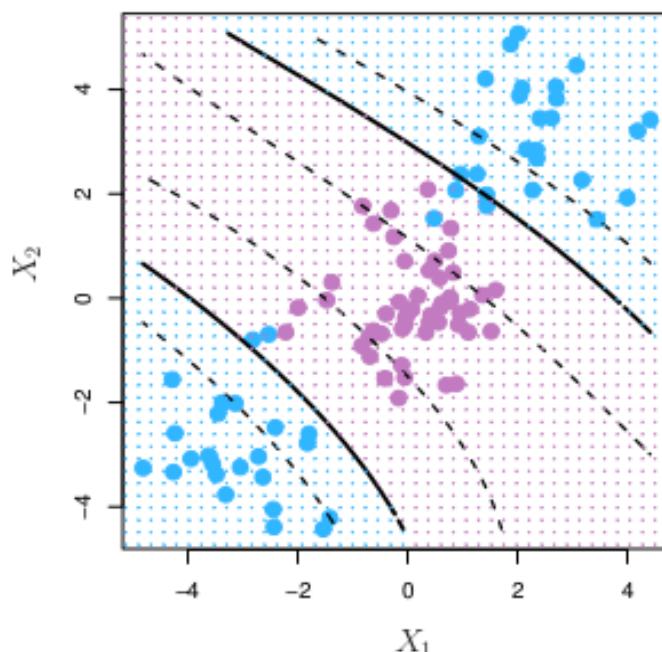
Esta transformación conlleva fronteras de decisión no-lineales en el espacio original.

## Polinomios cúbicos

Utilizando una transformación en forma de polinomios cúbicos se puede pasar de 2 a 9 variables. De esta forma el clasificador en el nuevo espacio soluciona el problema de buscar los patrones en el espacio con menor dimensiones.

En la siguiente gráfica se muestra el resultado de las fronteras de separación obtenidas por medio de una transformación en polinomios cúbicos., resultado del cálculo de la siguiente ecuación:

$$B_0 + B_1X_1 + B_2X_2 + B_3X_1^2 + B_4X_2^2 + B_5X_1X_2 + B_6X_1^2 + B_7X_2^3 + B_8X_1X_2^2 + B_9X_1^2X_2 = 0$$



Gráfica 8. Ejemplo de las fronteras de decisión no lineales obtenidas por medio de una transformación obtenida con polinomios cúbicos. Fuente: James et.al., 2013.

## 8.6. Kernel trick

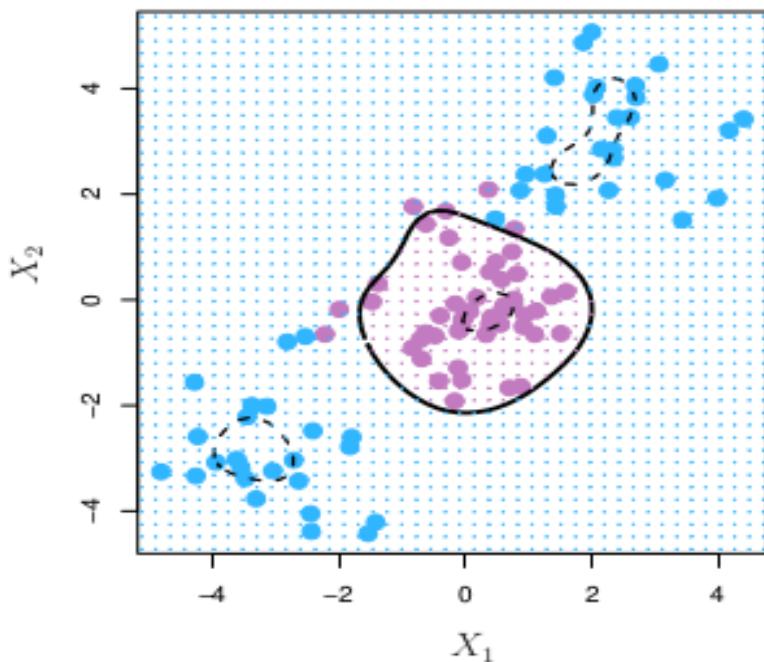
Las expansiones de las variables con polinomios, especialmente aquellos con grandes dimensiones, son computacionalmente costosos. Existe una solución más elegante y controlada de introducir no-linealidad que es utilizada en las máquinas vector de soporte por medio del uso de *kernels*. El *kernel* de un operador A denotado por *ker* es el conjunto de todos los vectores cuya imagen sea el vector nulo:

$$\ker A = \{ \vec{v} \in V : A \vec{v} = 0 \}$$

Los *kernels* se apoyan en concepto del producto vectorial de los vectores de soporte. Se trata de funciones que reciben dos vectores como parámetros. Uno de los *kernels* más utilizados es el de base radial que asume que el *feature space* es de altas dimensiones y se define con la siguiente función matemática:

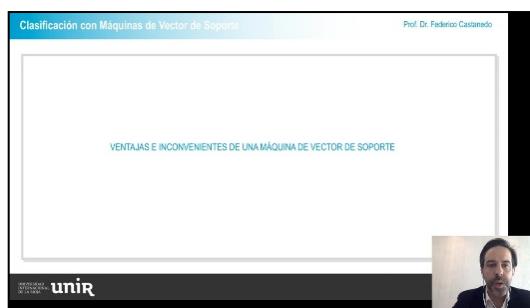
$$K(x_i, x'_i) = \exp(-\gamma \sum_{j=1}^p (x_{ij} - x'_{ij})^2)$$
$$y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) \geq M$$

El uso de *kernels* permite obtener fronteras de decisión no lineales por medio de transformaciones matemáticas sin necesidad de tener que realizar transformaciones con polinomios. En la siguiente figura se muestran las fronteras de decisión no lineales obtenidas por medio del uso de un *kernel* de base radial.



Gráfica. Ejemplo de las fronteras de decisión no lineales obtenidas por medio del uso de un *kernel* de base radial. Fuente: James et.al., 2013.

En este vídeo se van a repasar los conceptos de las máquinas de vector de soporte y se van a discutir las ventajas e inconvenientes de su aplicación.



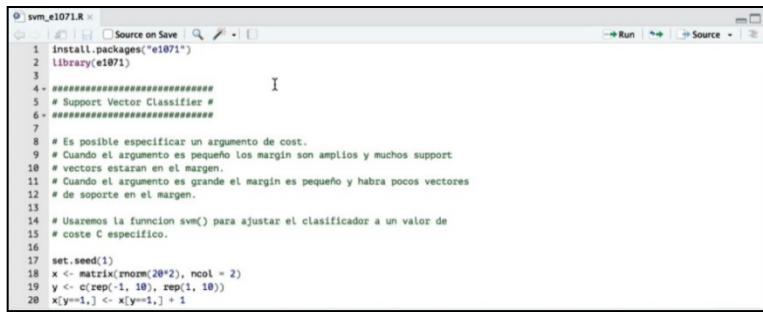
Ventajas e inconvenientes de una máquina de vector de soporte.

---

Accede al vídeo a través del aula virtual

---

A continuación, vamos a ver cómo entrenar una máquina de vector de soporte utilizando el paquete e1071 de R.



```
svme1071.R
1 install.packages("e1071")
2 library(e1071)
3
4 #####
5 # Support Vector Classifier #
6 #####
7
8 # Es posible especificar un argumento de cost.
9 # Cuando el argumento es pequeño los margin son amplios y muchos support
10 # vectores estarán en el margen.
11 # Cuando el argumento es grande el margin es pequeño y habrá pocos vectores
12 # de soporte en el margen.
13
14 # Usaremos la función svm() para ajustar el clasificador a un valor de
15 # coste C específico.
16
17 set.seed(1)
18 x <- matrix(rnorm(20*2), ncol = 2)
19 y <- c(rep(-1, 10), rep(1, 10))
20 x[y==1,] <- x[y==1,] + 1
```

Tutorial Entrenando una máquina de vector de soporte.

---

Accede al vídeo a través del aula virtual

---

## 8.7. Referencias bibliográficas

Hastie, T., Tibshirani R., Friedman, J. (2011). *The Elements of Statistical Learning*. Second edition. Springer.

James, G., Witten, D., Hastie, T and Tibshirani, R. (2013). *An Introduction to Statistical Learning with Applications in R*. Springer.

# Lo + recomendado

## No dejes de leer

### Support Vector Machines

Scikit learn. (s. f.). Support vector Machines.

Documentación del uso de las máquinas de vector de soporte utilizando la librería scikit-learn en Python

---

Accede a la página a través del aula virtual o desde la siguiente dirección web:

<http://scikit-learn.org/stable/modules/svm.html>

---

### Ejemplo de SVM en Python utilizando LibSVM, SVMLight y Pegasos

Jupyter nbviewer. (s. f.) Data Preprocessing.

Ejemplo de rendimiento y precisión de las librerías LibSVM, SVMLight y Pegasos en Python.

---

Accede a la página a través del aula virtual o desde la siguiente dirección web:

<http://nbviewer.jupyter.org/gist/nipunreddevil/5153583>

---

## No dejes de ver

### How Support Vector Machines work/how to open a black box

Explicación del funcionamiento de los máquinas de vector de soporte por Brandon Rohrer, *data scientist* en Facebook.

```
>>> from sklearn import svm  
>>> X = [[0, 0], [1, 1]]  
>>> y = [0, 1]  
>>> clf = svm.SVC()  
>>> clf.fit(X, y)
```

---

Accede al vídeo a través del aula virtual o desde la siguiente dirección web:

<https://www.youtube.com/watch?v=-Z4aojj-pdg>

---

## A fondo

### Support-Vector Networks

Cortes, C. Y Vapnik, V. (1995). Support-Vector Networks. *Machine Learning*, 20, 273-297.

Artículo de Vladimir Vapnik sobre las máquinas de vector de soporte.

---

Accede al artículo a través del aula virtual o desde la siguiente dirección web:

[http://image.diku.dk/imagecanon/material/cortes\\_vapnik95.pdf](http://image.diku.dk/imagecanon/material/cortes_vapnik95.pdf)

---

## Bibliografía

Corral, R. R. (2019). Análisis del Desempeño de Sistema de Detección de Señal SSVEP Utilizando Clasificadores Árbol Simple y Máquina de Vectores de Soporte. *ReCIBE*, 1-23.

# Test

1. Cuáles de las siguientes afirmaciones sobre las máquinas de vector de soporte son ciertas:
  - A. Su objetivo es buscar un plano que separe las clases en *feature space*.
  - B. Modifican la definición de separación.
  - C. Mejoran y enriquecen el *feature space* original.
2. Cuáles de las siguientes afirmaciones sobre los hiperplanos son ciertas:
  - A. Un hiperplano solo se define para 2 dimensiones.
  - B. En un espacio de 2 dimensiones el hiperplano es una recta.
  - C. Un hiperplano está definido para  $n$  dimensiones.
3. El vector normal:
  - A. Puede apuntar a una dirección ortogonal al hiperplano.
  - B. Debe apuntar a una dirección ortogonal al hiperplano.
  - C. No requiere ningún tipo de ortogonalidad.
4. Cuáles de las siguientes afirmaciones son ciertas:
  - A. Para una serie de datos de entrenamiento de un problema binario existen múltiples hiperplanos posibles.
  - B. Solo existe un hiperplano posible para cada uno de los conjuntos de datos.
  - C. Los hiperplanos hay que crearlos únicamente para realizar predicciones.

- 5.** Indique las afirmaciones ciertas sobre un *maximal classifier* en una clasificación binaria:
- A. Proporciona la mayor diferencia (*gap*) entre las instancias de cada uno de los ejemplos.
  - B. Proporciona información sobre cuantos márgenes de mejora existe.
  - C. Habitualmente los datos no son fácilmente separables por una recta y el modelo falla.
- 6.** De las siguientes afirmaciones *soft-margin* indique cuáles son correctas:
- A. Reduce de forma significativa el margen disponible para trabajar.
  - B. Utiliza desarrollo software para poder ejecutarse.
  - C. Permite separar puntos que no son linealmente separables.
- 7.** Cuáles de las siguientes afirmaciones sobre la función de coste son apropiadas:
- A. Cuanto más grande es el parámetro de coste mayor es el margen.
  - B. Cuanto más pequeño sea el parámetro de coste mayor es el margen.
  - C. La función de coste sirve para determinar el balance coste-beneficio de utilizar un modelo de clasificación.
- 8.** La expansión en forma de polinomios:
- A. Tiene un alto coste computacional.
  - B. Permite obtener fronteras de decisión no lineales sobre el espacio original.
  - C. Ninguna de las anteriores es correcta.
- 9.** Cuáles de las siguientes afirmaciones son una ventaja de los *kernels*:
- A. Permiten utilizar un menor volumen de información.
  - B. El coste computacional es menor al utilizar *kernels*.
  - C. Los *kernels* permiten obtener fronteras de decisión no lineales.

**10.** Cuáles de las siguientes afirmaciones sobre los *kernels* son ciertas:

- A. Se apoyan en el concepto de producto vectorial.
- B. Se trata de funciones que reciben dos vectores como parámetro.
- C. Se trata de funciones que reducen la información y pueden recibir cualquier parámetro.