

Tipos de aprendizaje y modelado de machine learning clásico

Introducción a la Inteligencia Artificial: Machine Learning



Tipos de aprendizaje y modelado de machine learning clásico

A lo largo de este tema vamos a estudiar los algoritmos del **aprendizaje supervisado** y los del **aprendizaje no supervisado**, así como las características de cada uno de ellos. Y es que los objetivos principales marcados para este fastbook son dos:

- Ser capaces de trabajar con datos tabulados para desarrollar algoritmos de machine learning.
- Saber entrenar y evaluar los algoritmos de clasificación, regresión, clusterización y reducción de dimensiones.

Con estas premisas, necesitaremos también ver de primera mano el uso del machine learning. Empezaremos con los modelos estadísticos, conocidos como *machine learning clásico*, pero que el término *clásico* no nos lleve a equívocos, ya que estos modelos no están en desuso. Muchos de ellos se siguen utilizando en producción, pero sobre todo para hacer esos análisis estadísticos que nos serán de gran utilidad antes de lanzarnos a deep learning.

 Machine learning

 Aprendizaje supervisado

 Aprendizaje no supervisado

 Conclusiones

 Recursos útiles

Machine learning



Para entender los algoritmos de machine learning, tenemos primero que ser capaces de distinguir entre sus tipos. Pues bien, nos encontramos con dos modelos de machine learning según su **uso de los datos (su aprendizaje)**:

1

Mediante el aprendizaje supervisado

El algoritmo aprende de un set de características (*inputs*) que se corresponden con una clase o valor (*labels*). El objetivo del algoritmo es aprender las relaciones entre los distintos *inputs* con sus *labels* correspondientes. Una vez aprendidas estas relaciones, se pueden usar para hacer predicciones en datos nuevos, los cuales no tienen por qué estar etiquetados. Los principales tipos de algoritmos basados en aprendizaje supervisado son dos: de clasificación o de regresión.

- **Clasificación:** se utiliza cuando queremos distinguir entre grupos de datos (por ejemplo, verdadero/falso, lluvia/sol/nublado).
- **Regresión:** cuando queremos predecir un valor numérico (por ejemplo, precios, cantidad de proteínas en una molécula...).

1

Mediante el aprendizaje no supervisado

El algoritmo no tiene un objetivo claro de sus *outputs*. Estos algoritmos no disponen de un *label* y los usaremos como métodos de exploración para entender mejor nuestros datos o generar agrupaciones sin ser clases fijas. Los principales son los algoritmos de clusterización y de reducción de *features*.

- **Clusterización:** se usa cuando queremos agrupar datos sin estar seguros si pertenecen a un grupo fijo (por ejemplo, el tipo de moléculas en un análisis de población o de tipo de clientes...). Es parecido a la clasificación.
- **Reducción de features:** se emplea cuando un dato individual puede tener decenas, centenas o incluso miles de características. Un dato con tantos valores es difícil de visualizar y es posible que no aporte para el problema que queremos resolver. Por ejemplo, la altura de un alumno no nos ayuda a entender si es más probable que apruebe o no una asignatura. En casos complejos, es conveniente tener un algoritmo no supervisado para entender mejor la composición de nuestros datos.

El objetivo de machine learning entonces es doble, por un lado, busca cómo entender, analizar y/o visualizar datos y, por otro, crear modelos que puedan dar un valor como producto final

Aprendizaje supervisado



Llamamos modelos de aprendizaje supervisado a aquellos en los que podemos definir el *output* esperado por el modelo.

Se denominan *supervisados* porque podremos indicar al algoritmo cómo de bien o mal está resolviendo el problema.

Para poder hacer uso del aprendizaje supervisado tendremos la necesidad de tener **datos etiquetados (labeled)**. Esto quiere decir, que una de las características de nuestro modelo va a ser su objetivo. Por ejemplo:

- Los precios de casas, donde podemos tener características de la casa como su ubicación, dimensiones, calidad energética, etc., e intentaremos estimar su precio.
- Los tipos de películas con información como el reparto, la compañía productora, dirección, duración, etc., para determinar si es una película de terror, comedia o drama.

- La puntuación en la reseña de un restaurante según su tipo de comida, los comentarios que recibe y su evolución a lo largo del tiempo.

Clasificación

Dentro del aprendizaje supervisado, la clasificación se encarga del desarrollo de modelos que sean capaces de distinguir entre **clases en nuestros datos**. Estas clases pueden ser **valores numéricos o de texto** (0/1/2 o izquierda/derecha).

Algunos de los ejemplos más comunes son los siguientes:

- Marcar si un email es de spam o no.
- Tipo de clima que hará al día siguiente (lluvia/sol/nublado).
- Clasificación de animales en sus familias (gatos/perros o mamíferos/anfibios/...).

Internamente, el modelo puede transformar estas clases en números (por ejemplo: 'Spam' se marca como 0, 'No spam' se marca como 1), pero estos valores no implican una jerarquía y no cambia el resultado si se entrena con otro mapa de valores.

Un modelo de clasificación, por lo tanto, recibirá un dato con una serie de características y devolverá su clase. Para entrenar, necesitaremos de datos etiquetados para que aprenda de ejemplos qué tipos de datos se asocian a una clase u otra.

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.25		S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282	7.925		S
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	53.1	C123	S
5	0	3	Allen, Mr. William Henry	male	35	0	0	373450	8.05		S
6	0	3	Moran, Mr. James	male		0	0	330877	8.4583		Q
7	0	1	McCarthy, Mr. Timothy J	male	54	0	0	17463	51.8625	E46	S
8	0	3	Palsson, Master. Gosta Leonard	male	2	3	1	349909	21.075		S
9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27	0	2	347742	11.1333		S
10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14	1	0	237736	30.0708		Q
11	1	3	Sandstrom, Miss. Marguerite Rut	female	4	1	1	PP 9549	16.7	G6	S
12	1	1	Bonnell, Miss. Elizabeth	female	58	0	0	113783	26.55	C103	S
13	0	3	Saundercock, Mr. William Henry	male	20	0	0	A/5. 2161	8.05		S
14	0	3	Andersson, Mr. Anders Johan	male	39	1	5	347082	31.275		S
15	0	3	Vestrom, Miss. Hulda Amanda Adolfinia	female	14	0	0	350406	7.8542		S
16	1	2	Hewlett, Mrs. (Mary D Kingcome)	female	55	0	0	248706	16		S
17	0	3	Rice, Master. Eugene	male	2	4	1	382652	29.125		Q
18	1	2	Williams, Mr. Charles Eugene	male		0	0	244373	13		S
19	0	3	Vander Planke, Mrs. Julius (Emilia Maria Vandemoortele)	female	31	1	0	345763	18		S
20	1	3	Masselmani, Mrs. Fatima	female		0	0	2649	7.225		C
21	0	2	Fynney, Mr. Joseph J	male	35	0	0	239866	26		S

Este es el *dataset* de pasajeros en el Titanic, donde se podría usar ‘Survived’ como *label* y el resto de las columnas como características. Fuente: Quantum-Like Sampling, de Andreas Wichert.

A continuación, vamos a revisar algunas de las **técnicas** más utilizadas para solucionar retos de clasificación dentro de machine learning.

1

Árboles de decisión

El objetivo de un árbol de decisión es ser capaz de seguir un camino o grafo respondiendo a preguntas de sí y no, llegando hasta un punto en donde obtendremos nuestra respuesta (clase). **Los árboles de decisión están compuestos por nodos.**

Por tanto, estos son los **componentes principales** de un árbol de decisión:

- **Los nodos**, o puntos de decisión donde se va a plantear una pregunta de sí o no. Dependiendo de la respuesta, el dato seguirá una rama u otra. Por ejemplo, en el caso de predicción del tiempo podríamos tener: “¿Es otoño?” “¿Es la humedad superior al 20%?”

- **Las ramas**, que son las rutas que nos van a llevar entre un nodo y el siguiente. Dependiendo de si responde a un nodo con un “sí”, iremos por su rama correspondiente o por la rama indicada con un “no”.
- **Las hojas**, que sería un nodo particular caracterizado por no tener más ramas. Al llegar a una hoja tendremos nuestra clase.

Para saber qué nodos generar y con qué jerarquía, el algoritmo sigue y repetirá los siguientes **pasos**:

Elección del nodo root o nodo de entrada

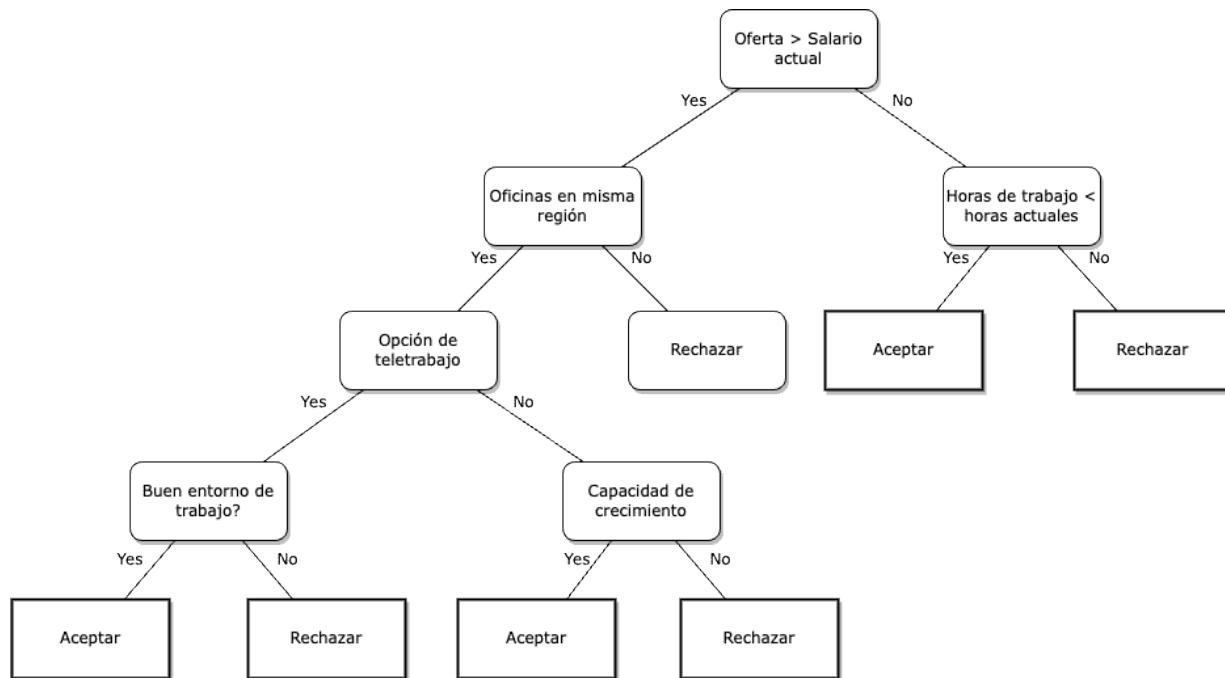
Esto se hace seleccionando la variable que mejor separa los datos entre las clases necesarias según un cálculo de impureza con el algoritmo *gini*, aunque existen otros algoritmos para determinar la impureza de un nodo. Y el *input* con menor grado de impureza se selecciona como el nodo *root*.

Creación de nodos internos

Por cada separación de datos que se cree, según nuestro nodo *root*, el algoritmo repetirá el proceso para obtener una separación más fina de los datos. Esto se continúa hasta llegar a un criterio de parada como la profundidad del árbol, el tamaño de las hojas, el límite de impureza, etc.

Asignación de nodo hoja

Cuando un nodo supera uno de los criterios mencionados, el modelo asigna ese nodo como hoja. El nodo hoja comprueba cuál es la *label* preferente en él y lo asigna como resultado final.



Este es un ejemplo de árbol de decisión. Fuente propia.

Los árboles de decisión son un modelo que puede aportar gran valor en poco tiempo por tres motivos:



son un buen primer acercamiento para entender qué características son más relevantes en nuestros datos;

- resultan fáciles de entender, incluso si no se sabe cómo se han generado;
- y pueden ser modelos productivos que solucionan la toma de decisiones complejas.

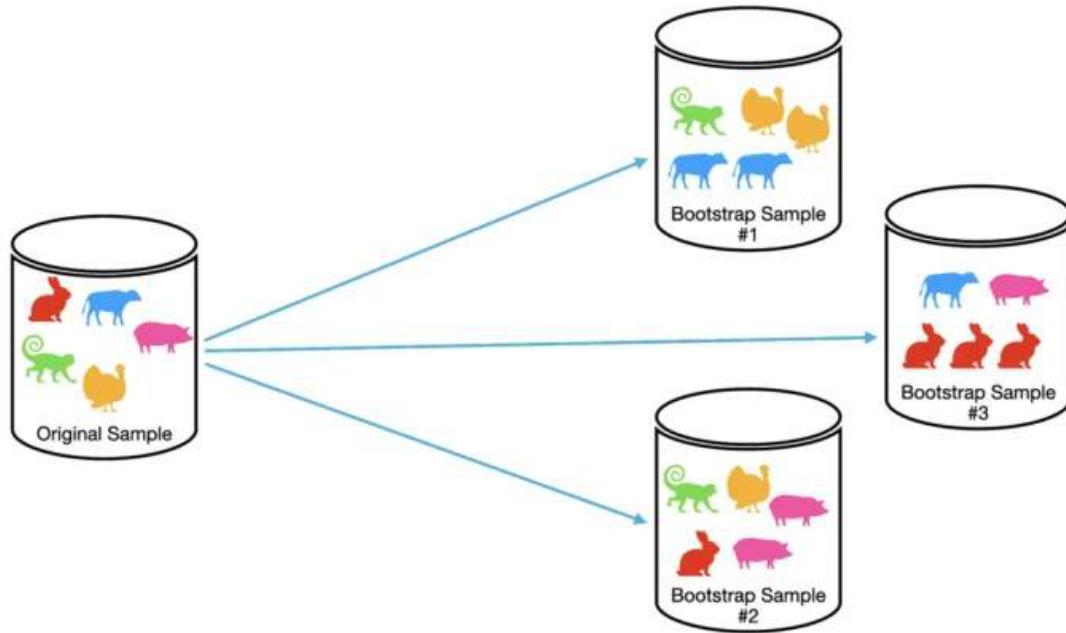
2

Random forest

Random forest es un tipo de modelo denominado *ensemble*, en el que se usan dos o más modelos para obtener un mejor resultado que los modelos que lo componen. Random forest se puede aplicar sobre varios modelos bases, pero uno de los más comunes es como un *ensemble* de árboles de decisión a los cuales se le va a aplicar una aleatoriedad en sus datos de entrenamiento.

Una de las formas en que se aleatorizan los datos de entrenamiento es con su muestreo. Llamamos muestreo a la selección específica de los datos. El método usado en random forest se llama *bootstrap*, en el cual se hace un muestreo con sustitución.

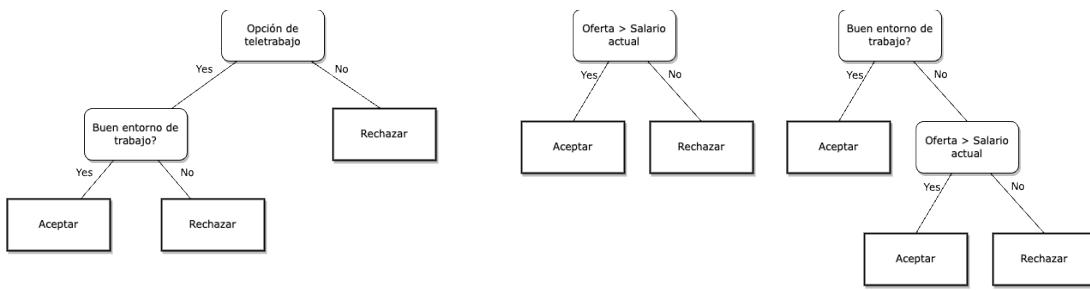
Un muestreo con sustitución se basa en coger datos del dataset de uno en uno y la selección de este dato se hace de forma aleatoria. Dicho proceso se repite hasta tener un dataset del mismo tamaño que el original. Sin embargo, cada selección es aleatoria, lo que significa que un mismo dato se puede escoger más de una vez. Esto hace que el nuevo dataset se distinga del original al tener datos que falten o se repitan.



Este sería un ejemplo de muestreo Bootstrap. Fuente: Insidelearningmachines.com

Además de tomar datos aleatorios, tendremos información parcial en cada *step* de entrenamiento. A la hora de crear un nodo, en vez de considerar todas las variables, se escogerá un *subset* de las variables para formar el nodo. Esto hace que el árbol final no sea óptimo y que dos árboles que se entrenen con el mismo conjunto de datos puedan tener estructuras distintas. Aunque parezca que esto sea contraproducente, ayuda a que el *ensemble* obtenga distintos puntos de vista al tener árboles que se centran en la distinción de datos desde un punto diferente.

Otro aspecto destacable de la técnica de *random forest* es la toma de decisión final, puesto que tendremos dos o más modelos que puede que no clasifiquen cada dato de la misma forma. El método más común es por voto mayoritario (la clasificación que indiquen el mayor número de modelos).



Aquí vemos un ejemplo de *random forest*. Fuente: elaboración propia.

Recuerda: *random forest* permite mejorar los resultados sobre árboles de decisión cuando las características tienen una importancia similar. A base de forzar a cada árbol para que se fije en muestras de datos distintas, consigue que ninguna característica tome el control de las decisiones. Esto hace que los nuevos datos, que podrían no estar representados en entrenamiento, se puedan clasificar correctamente.

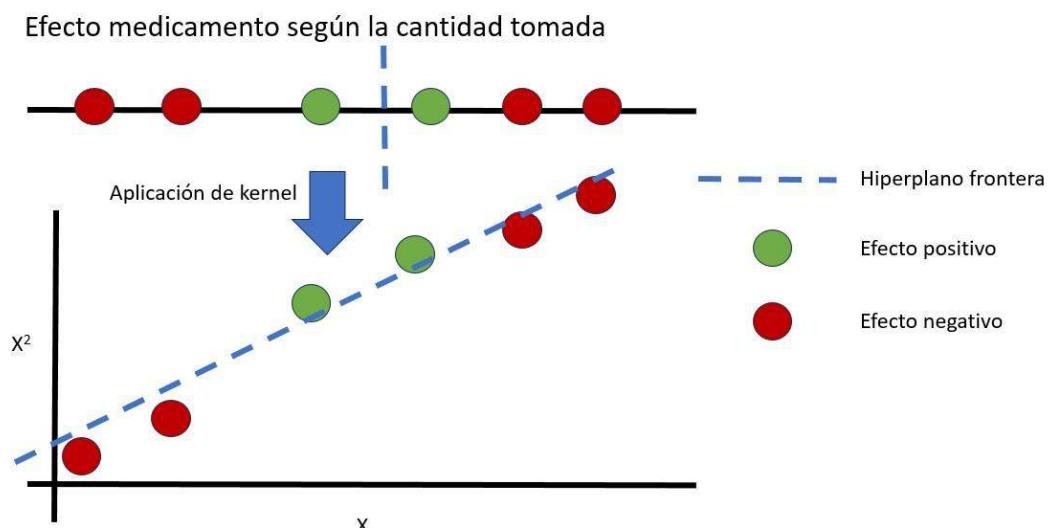
3

Support vector machine

Support Vector Machine (SVM) es un tipo de modelo que busca ser capaz de separar nuestras clases creando un **hiperplano frontera**. Para entender un hiperplano, supongamos que tenemos datos de una dimensión (p. ej., la edad de personas) con dos clases (p. ej., jubilado o jubilada/ no jubilado o no jubilada). El hiperplano en este caso será el valor margen que separa las dos clases (p. ej., 65 años).

Un hiperplano en datos con dos características será una línea, para tres características un plano, etc. SVM busca el hiperplano que maximice el margen de las clases o, dicho de otro modo: buscar el plano que tenga mayor distancia entre el punto más cercano al plano de cada clase. Esto permite tener un margen de espacio para errores en nuevas muestras.

Este método por sí solo no es una gran aportación y tiene una gran **limitación**. Supongamos la efectividad de una nueva medicina donde tomar muy poca cantidad de ella no tiene efecto, pero tomar demasiada puede generar complicaciones. En este caso, no existe un único valor donde separar los valores. Para evitar este problema se aplicarán los *kernels* (algoritmos que permiten elevar a mayores dimensiones los datos), lo que permitirá encontrar un plano que los separe. Uno de los kernels más comunes es el **cuadrático**, el cual eleva al cuadrado cada dato para aumentar sus dimensiones.



Este es un ejemplo de clasificación de dosis en SVM. Fuente: elaboración propia.

SVM, sin duda, es un algoritmo con muchos casos de usos interesantes. La capacidad de aumentar las dimensiones de los datos le permite explorar espacios que no se exploran con métodos como los árboles de decisión. Este algoritmo es un primer paso para ver cómo haciendo pequeñas operaciones sobre los datos se pueden obtener mejores resultados (algo que será muy relevante en deep learning).

4

Regresión logística

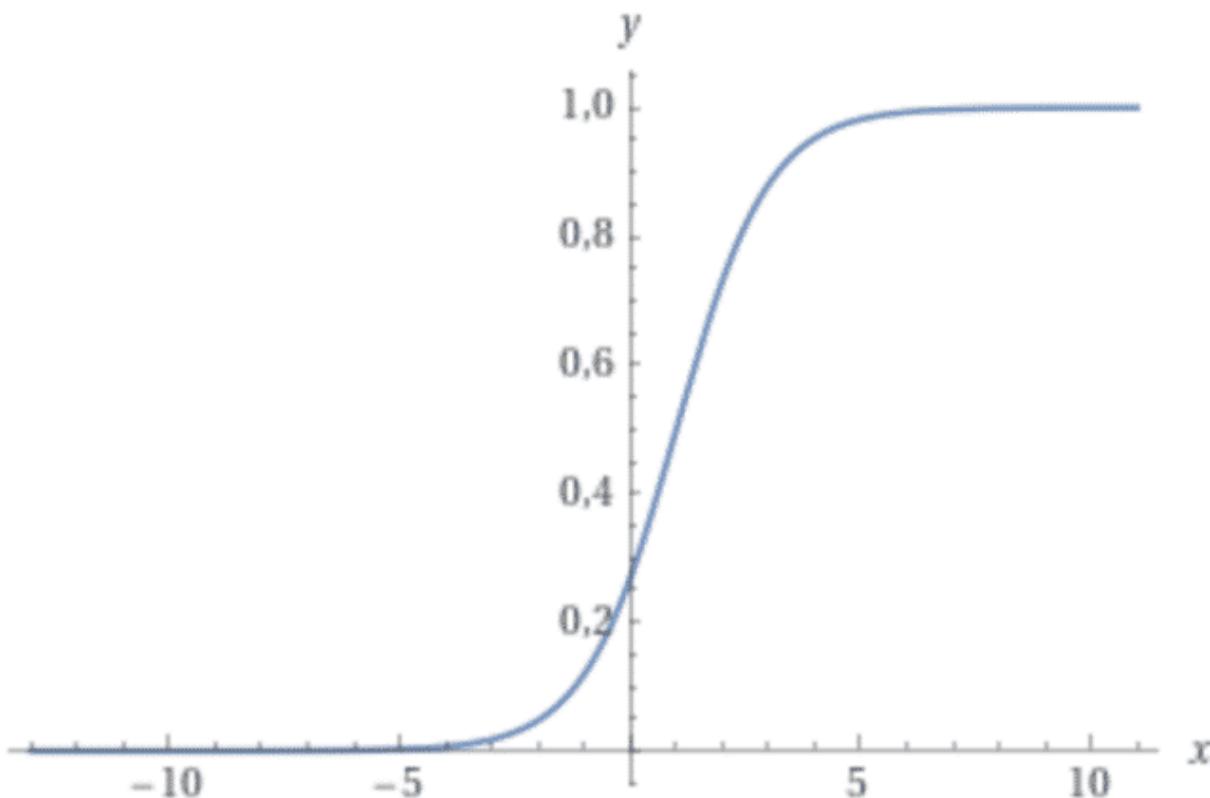
A pesar de su nombre, la regresión logística tiende a ser usada como un modelo de clasificación. Una regresión logística intenta calcular la probabilidad de que un valor pase de una clase a otra mediante una ecuación logística.

La regresión logística se representa con la siguiente ecuación:

$$p(x) = \frac{1}{1+e^{-(x-\mu)/s}}$$

Donde μ indicará el centro por donde pasa el cambio de clase; s cómo de lento será el paso de una clase a otra; y $p(x)$ es el valor de la clase. Nuestro modelo aprenderá los valores de μ y s para distinguir entre nuestras clases.

Para una visualización de esta ecuación, disponemos de la siguiente gráfica:



El modelo de regresión logística es exclusivamente binario y necesita que sus clases sean numéricas (1 y 0 normalmente). Pero es un gran algoritmo cuando las clases son suaves, cuando un valor puede ser un porcentaje de una clase y un porcentaje de la otra. Esto se consigue con los valores centrales, que indican un mix entre las dos clases.

El algoritmo de regresión logística será de gran utilidad en deep learning, como veremos a lo largo del temario de este curso.

5

Evaluación de modelos de clasificación

Este conjunto de algoritmos nos ofrece una base de cómo intentar solucionar nuestros problemas de clasificación, pero un factor importante será el poder **compararlos** unos con otros. Para ello, usaremos las **métricas de clasificación** (accuracy, precisión...).

Para entender las métricas de clasificación, pensemos en una prueba de una enfermedad, esta prueba cuenta con dos clases: positiva (P), si se ha contraído la enfermedad; y negativa (N), en el caso de no haber sido así. Nuestro modelo puede asignar una de estas dos clases y ser correcta o incorrecta, de tal forma que...

- Una persona que padezca la enfermedad y cuya prueba indique que la ha contraído se denominará **true positive (TP)**.
- Una persona que no padezca la enfermedad, pero la prueba indique que sí la ha contraído se denominará **false positive (FP)**.
- Una persona que padezca la enfermedad, pero la prueba indique que no la ha contraído se denominará **false negative (FN)**.
- Una persona que no padezca la enfermedad y cuya prueba indique que no la ha contraído, se denominará **true negative (TN)**.

Aquí podemos ver que un algoritmo puede cometer **2 tipos de errores (FP y FN)**. Por lo que es necesario ver cómo lo evaluamos.

Así que volvamos a las métricas y cómo miden. Normalmente, solemos hablar de tres como las principales: *accuracy*, *recall* y *precision*. Vamos a estudiarlas a través de nuestro ejemplo.

ACCURACY

RECALL

PRECISION

Del **conjunto de resultados**, mide qué porcentaje son los aciertos: cuál es la proporción de aciertos sobre todas las predicciones realizadas. Un *accuracy* perfecto representa que no se ha cometido ningún tipo de fallo, pero un *accuracy* imperfecto no nos indica dónde hay errores.

ACCURACY

RECALL

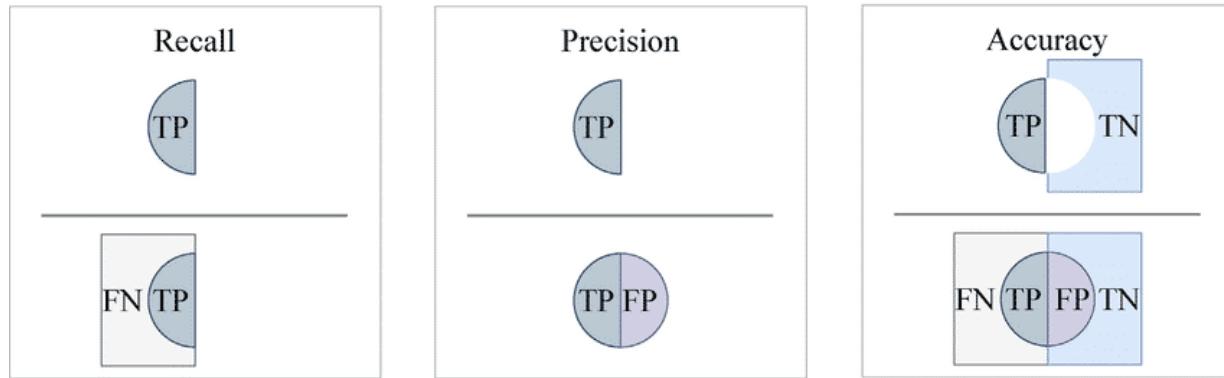
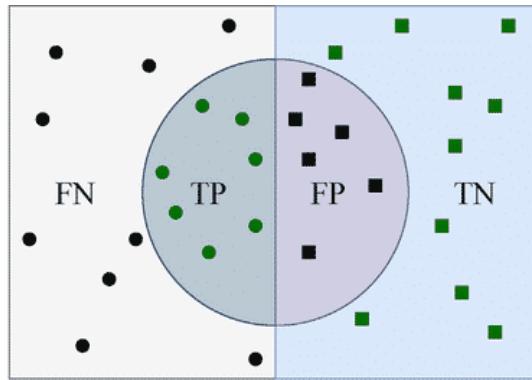
PRECISION

Del **conjunto de muestras positivas**, mide cuántas han sido detectadas. Esta métrica nos mostraría de todos los casos positivos, cuántos son detectados. Por tanto, es importante en pruebas, por ejemplo, de detección de enfermedades contagiosas, ya que es más relevante haber encontrado todos los casos positivos que detectar algún falso negativo. Un *Recall* perfecto indicará, por ejemplo, todas las clases identificadas como 'Positivo' que se encuentran, pero no necesariamente las predicciones 'Positivo' serán correctas.

ACCURACY	RECALL	PRECISION
<p>Del conjunto de respuestas positivas, mide cuántas son 'True', es decir, esta métrica nos muestra con cuánta seguridad (fiabilidad) un resultado positivo es realmente positivo. Ignora por completo los negativos, pero en ciertos casos contabilizar los resultados negativos no es el foco central de la clasificación. Por ejemplo, en la detección de una enfermedad grave, es mejor tener falsos negativos (FN) y repetir la prueba que tener un positivo sin la seguridad de que así sea. Una precisión perfecta indicará que todas las predicciones 'Positivo' son correctas, pero no necesariamente se habrán encontrado todas las clases 'Positivo'.</p>		

Recuerda dos detalles importantes:

- En español tendemos a traducir Accuracy como precisión, al igual que *precision*. Al presentar resultados en español, debemos tener cuidado con qué métrica estamos representando.
- Las métricas que acabamos de ver son la referencia para cualquier evaluación de clasificación binaria.



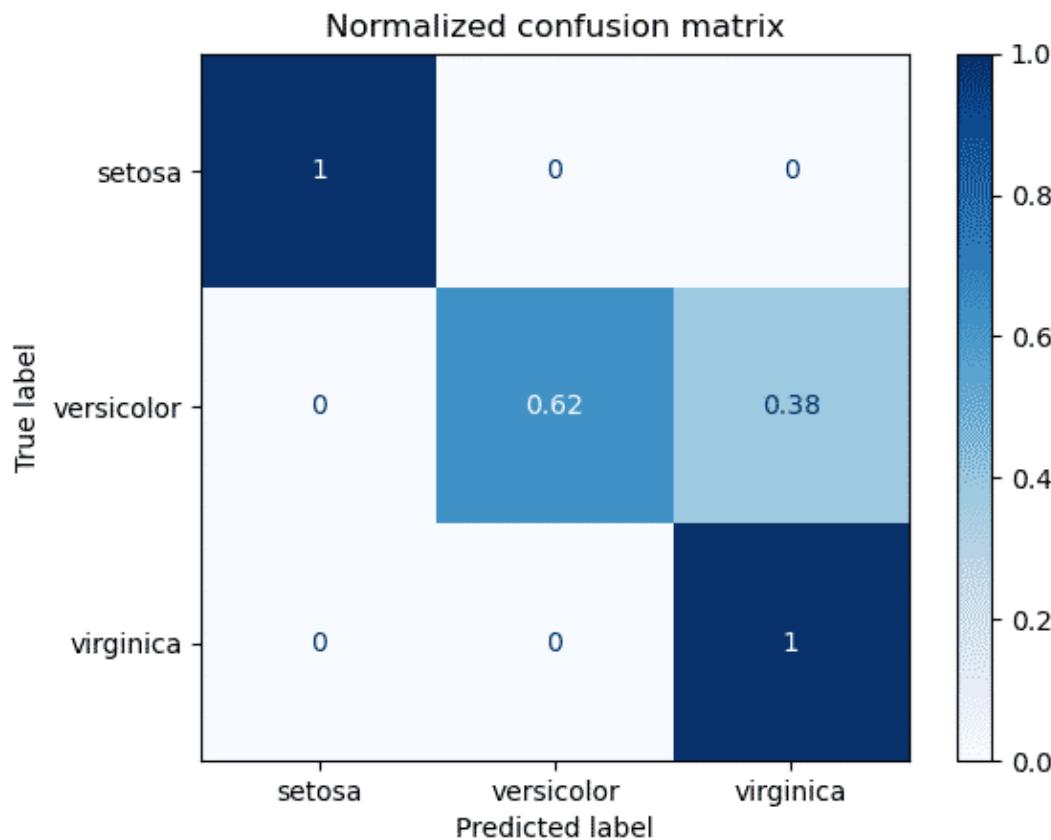
Representación de las métricas de clasificación de los resultados de las pruebas de enfermedad. Fuente: *Overview of Machine Learning, Part 1*, de varios autores.

Para una **clasificación multiclasa**, las métricas se pueden mantener centradas en cada una de las clases. En una clasificación de 'sol/lluvia/nublado', podemos hablar de la precisión de 'nublado' o el *accuracy* general del modelo. Sin embargo, hay formas de poder evaluar los resultados específicamente para multiclasa:

- **F1-score.** Similar al *accuracy* *F1-score*, intenta evaluar el comportamiento general del modelo al clasificar, pero centrándose más en los errores del modelo. Esto se calcula mediante la siguiente fórmula:

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

- **Matriz de confusión.** Una representación gráfica de cómo las clases reales son clasificadas. En un eje tendremos las clases reales, mientras que en el otro tendremos las predichas. Esto permite ver qué clases se confunden en el modelo.



Matriz de confusión en la que se ve que los errores se deben a la clase versicolor.
 Fuente: *Plot confusión matrix*, publicado en Scikit-learn.org

Estas métricas que hemos estudiado nos ayudarán a poder **comparar** entre los distintos tipos de modelos de clasificación. En su conjunto nos mostrarán los detalles de cómo está funcionando nuestro modelo y, para casos donde una clase sea más importante que otra, podremos centrarnos en la métrica más relevante.

Regresión

Mientras que en la clasificación intentábamos meter nuestros datos en cajas, con el método de regresión vamos a **buscar un valor más continuo y jerárquico**.

Llamamos modelos de regresión a aquellos modelos que intentan relacionar los valores de *input* y una magnitud. Esta magnitud pueden ser precios, cantidades, rankings, etc.

Veamos ahora las principales **diferencias entre clasificación y regresión**.

- Relación entre valores:** al igual que en clases podíamos indicar Spam (0) y No Spam (1), pero no se indicaba que uno era superior al otro, en regresión sí existe esa relación. Por ejemplo, una magnitud 5 será superior a una magnitud 4 e inferior a una 6.
- Continuidad:** los modelos de regresión por lo general entenderán que existe una continuidad de los datos. Por lo tanto, incluso si nuestras magnitudes son cantidades enteras (1, 2, 3) nuestro modelo puede darnos valores intermedios (0.5, 1.3, 4.2). Hay métodos para forzar a cantidades enteras, pero suelen venir posterior al modelo (el redondeo).

Entendidas las diferencias, a continuación, vamos a revisar algunas de las técnicas más conocidas para **solucionar los problemas** de regresión dentro del machine learning.

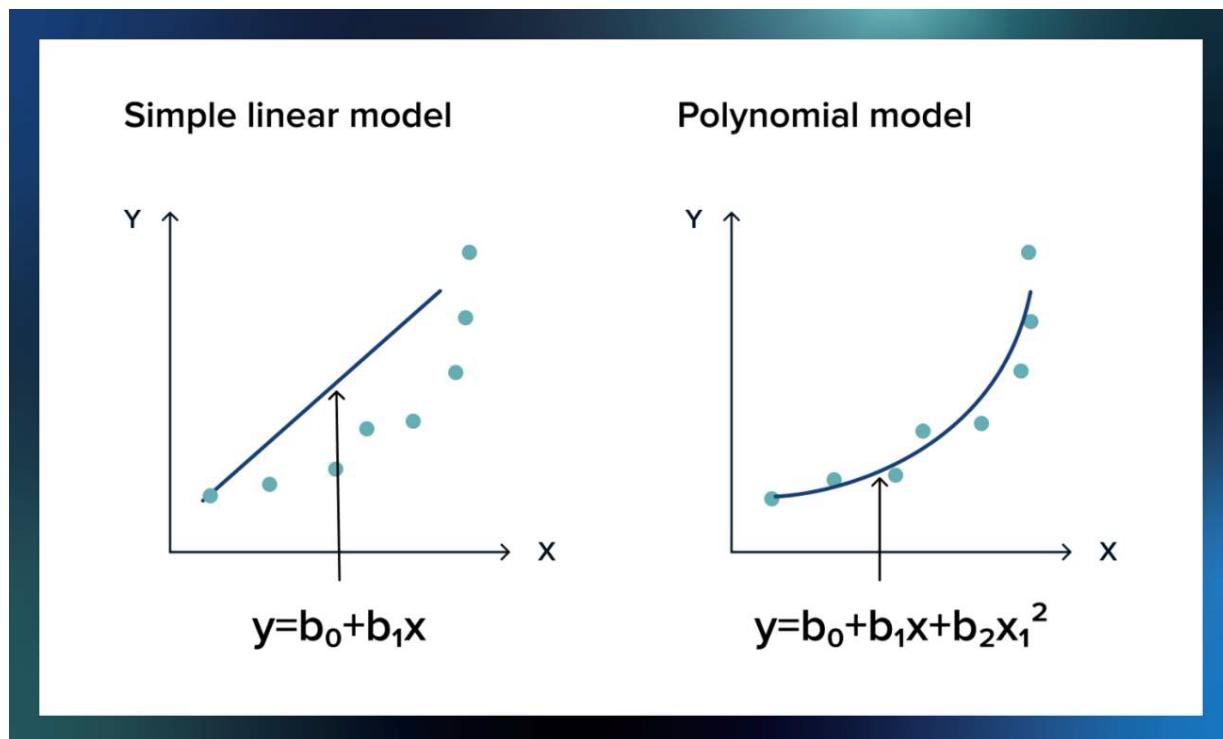
1

Regresión lineal

La regresión lineal es un algoritmo que representa su salida (o variable dependiente) como una función de suma de peso de sus *inputs* (predictores o variables independientes). La base fundamental es que hay una **relación lineal entre las variables** independientes y la dependiente.

En su versión más simple, tendremos una **regresión lineal simple o de primer grado** en la que, después de entrenar, tendremos una función en la que todos los inputs son multiplicados por un peso dedicado y se sumarán para dar el valor final (comúnmente también se añadirá un factor independiente constante). Para casos más complejos tendremos **regresiones polinomiales** donde, además de sus pesos lineales, se les sumarán pesos a los inputs elevados al cuadrado, tercer grado, cuarto, etc.

Para poder entrenar estos algoritmos se usan **ecuaciones de distancia media** (normalmente *mean squared error*) y se hace un trabajo de **optimización** para tener el menor error posible.



Regresión polinómica, donde los pesos b son los valores que se entrena para generar la función. Fuente: *Polynomial regression analysis*. Serokell.io

La regresión lineal (y sus versiones polinómicas) es un acercamiento rápido y sencillo. Estas regresiones muestran cuán fuerte es la **relación de los datos con sus labels**. Incluso si el algoritmo no obtiene resultados productivos, proporciona una visión clara de la relación entrada-salida.

2

LASSO regression

El algoritmo LASSO (*Least Absolute Shrinkage and Selection Operator*) es un algoritmo de regresión lineal que penaliza pesos bajos. En la regresión lineal es raro ver pesos exactos a cero, ya que toda variable tiene un pequeño aporte a la variable dependiente (aunque sea ruido). La regresión LASSO fuerza una conversión a 0 los pesos muy pequeños.

Esta característica del algoritmo LASSO tiene **dos ventajas** principales:

- La eliminación de ruido**, al no tener en cuenta pequeños aportes que posiblemente no generalicen.
- La selección de variables**: si en entrenamiento una variable tiene aporte 0, podemos eliminarla de nuestros datos de entrada.

LASSO es un algoritmo de doble uso, como modelo de regresión y de reducción de dimensiones. Ser capaz de reducir el número de dimensiones es importante en muchos casos, ya que reduce ruido en los datos. Esto hace que LASSO tienda a ser más preciso que la regresión lineal; aunque no siempre será una mejora, puesto que algunas características menores pueden ayudar a ver las pequeñas diferencias entre labels.

3

Evaluación de modelos de regresión

En la evaluación de modelos de regresión, querremos ver cómo de cerca está la predicción del modelo con respecto al valor real. Al contrario que con clasificación, un error de regresión no se puede definir como acierto o fallo, si no un error de 500€ sería igual a uno de 3000€.

Aunque existe un abanico de **métricas para medir el error cometido** en los modelos de regresión, aquí mostramos dos de las más usadas:

Mean Absolute Error (MAE)

Error que expresa la distancia absoluta media de los errores. Es un cálculo simple que evalúa todos los errores de forma proporcional.

$$MAE = \frac{1}{N} \sum |y_{real} - y_{prediction}|$$

Root Means Squared Error (RMSE)

Expresa la distancia del fallo de forma cuadrática. Esto hace que grandes errores tengan un mayor peso que en MAE, lo que penaliza aún más fallar los errores de gran valor.

$$RMSE = \sqrt{\frac{1}{N} \sum (y_{real} - y_{prediction})^2}$$

La selección entre las dos métricas se debe a cuánto queremos penalizar los errores grandes. En un caso donde un error es proporcionalmente igual de grave, podemos usar MAE. En los casos en los que un error grande pueda llevar a mayores consecuencias, usaremos RMSE. A menor valor en cualquiera de ellos, más preciso será nuestro modelo.

Aprendizaje no supervisado



El segundo gran grupo de machine learning es el aprendizaje no supervisado. Al contrario que con el aprendizaje supervisado, no se tendrá un valor objetivo para cada dato, lo que significa que no podemos definir clases o hacer regresión de un valor.

El aprendizaje no supervisado es muy común en el **mundo empresarial** por dos motivos principales:

La falta de datos etiquetados

Un factor muy costoso en algunos casos. Hay campos en los que las clases son conocidas, pero no están etiquetadas.

Veámoslo con un ejemplo: una fábrica de juguetes hace fotos a todos sus productos antes de mandarlos a las tiendas; de entre ellas, una de cada millón tiene un defecto y queremos detectar ese defecto. Para poder tener dos clases (defectuoso/no defectuoso), requeriría revisar millones de imágenes y la desproporción de las clases sería enorme. Un modelo de aprendizaje no supervisado puede ayudar como primer acercamiento al problema para después poder etiquetar.

Making It Real

Es decir, aquellos casos en que el objetivo del modelo no es claro o ambiguo y que, además, serán más de estudio que de uso productivo. Esto se puede deber a que se tienen muchas características de un dato, pero se desconoce cuáles son útiles o cuáles serían las clases de partida.

Un ejemplo para entenderlo mejor sería una empresa que recopila información de los usuarios sobre sus hábitos de uso de una aplicación y quiere estudiar si es posible explotar esos datos de alguna forma.

Una diferencia importante entre los algoritmos del aprendizaje no supervisado y los del supervisado es el **uso de métricas**. En el aprendizaje no supervisado, no disponemos de labels al que comparar, por lo que no se pueden tener métricas que definen con total precisión cómo de bien o mal está trabajando un modelo. Sin embargo, sí que existen algunas métricas que se pueden usar para comparar resultados.

1

Clustering

Similar a la clasificación en aprendizaje supervisado, *clustering* intenta **agrupar el conjunto de datos** con el que estemos trabajando. Sin embargo, al contrario que con clasificación, no tendremos clases definidas. Para poder hacer agrupaciones de datos, lo haremos a través de su **similitud**. El número de grupos en muchos casos tendremos que definirlo nosotros, lo que puede llevar a un proceso de iteraciones hasta dar con el número de clústeres que se adapte a nuestras necesidades.

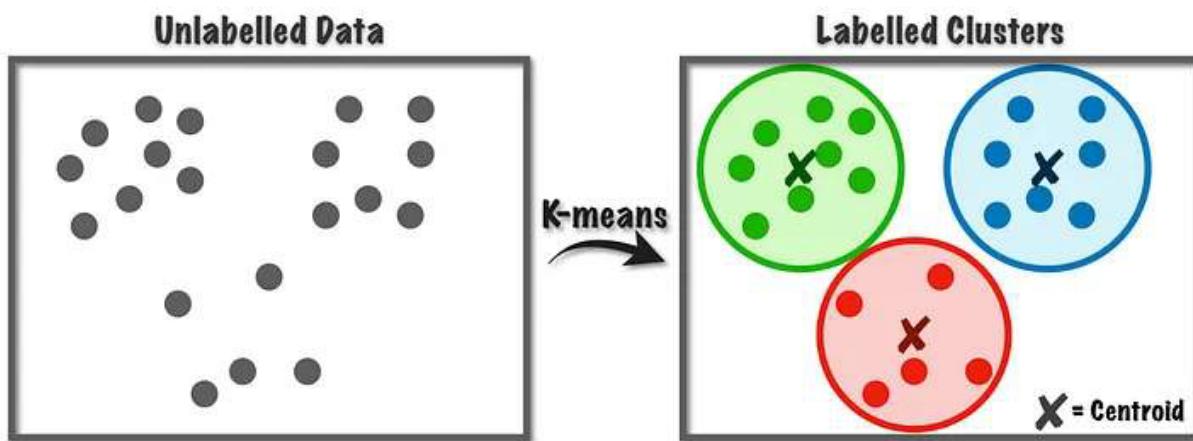
Uno de los casos de uso más representativos de este enfoque es la **agrupación de usuarios**. Un caso muy típico en aplicaciones como Spotify o Netflix, donde se recomiendan películas o música según los gustos de la persona. Esto se consigue a base de clusterizar a gente con gusto similares. Estos clústeres no son una garantía de clase real, pero ayudan a agrupar a los clientes.

Hay muchas aproximaciones basadas en clustering para resolver problemas de aprendizaje no supervisado. En este fastbook vamos a revisar dos de las **técnicas más utilizadas** en la actualidad: **K-means** y **DBSCAN**.

- **K-means**

K-means es uno de los algoritmos más conocidos en el ámbito de clustering. El algoritmo se centra en **agrupar los datos en un número determinado de grupos (K)**, el cual tendremos que asignar al inicializar el modelo.

Para conseguir estos grupos, el algoritmo plantea un número K de centroides, los cuales tendrán un valor aleatorio en cada uno de los inputs de los datos. A continuación, se agrupan los datos calculando su centroide más cercano. Por cada grupo, se obtiene la media de valores. Una vez calculada, se desplazan los centroides en dirección a la media calculada. Este proceso se sigue repitiendo hasta un número de iteraciones o hasta que el modelo no consiga mejorar, lo que se interpreta como una resolución óptima de los grupos.



Este es un ejemplo de K-mean. Fuente: *K-means: A Complete Introduction*, de Alan Jeffares. Towardsdatascience.com

Como se puede ver, tenemos un factor importante en el número de centroides K. Dependiendo del problema, puede que tengamos una estimación de cantidad de grupos necesario o no. Nuestro trabajo será jugar con este valor para obtener un resultado positivo.

En definitiva, K-means es un algoritmo capaz de encontrar agrupaciones de datos. En dos dimensiones estas agrupaciones pueden ser evidentes, pero cuando los datos tengan más dimensiones, encontrar agrupaciones puede ser de gran ayuda.

No solo eso, encontrar un número de clústeres puede ayudar a la hora de entender los datos. Si miles de datos se reducen a 3 o 4 clústeres, se pueden usar como simplificación para agrupar, recomendar o entender similitudes entre subconjuntos de los datos disponibles.

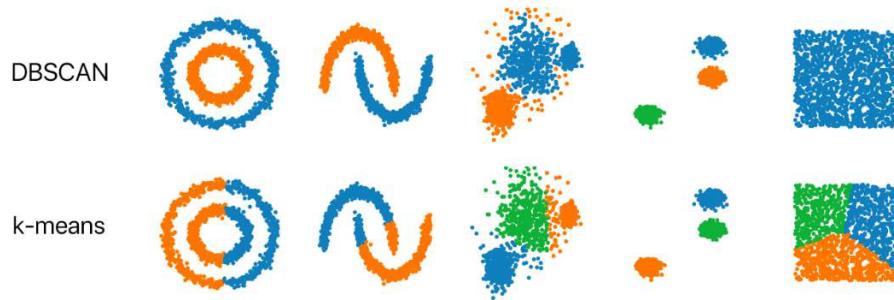
- **Density-based spatial clustering of applications with noise**

En casos donde los datos no se relacionan simplemente por cercanía o bien no podemos estimar el número de clústeres, se puede usar *Density-Based Spatial Clustering of Applications with Noise (DBSCAN)*. En vez de buscar núcleos de datos, DBSCAN **explorará agrupaciones por distancia**, haciendo cadenas de datos al estar asociados con sus vecinos. Esto se lleva a cabo incluso si hay datos más alejados de un centro del clúster.

Para poder ejecutar DBSCAN necesitamos entender varios conceptos, ¡toma nota!

- Punto central (core point):** un punto que tenga un mínimo de puntos a una distancia cercana (el número de puntos y la distancia de esos puntos tendremos que definirlo previamente).
- Punto frontera (border point):** punto que esté dentro del área de un punto core, pero no sea punto core por sí mismo.
- Ruido (noise o outlier):** puntos que no entran en ninguna de las otras dos categorías.
- Conexión por densidad:** dos puntos están conectados por densidad si hay una cadena de puntos core que los une, lo que hace que sean del mismo clúster.
- Clúster:** conjunto de puntos (core y frontera) conectados densamente. Más de un punto core puede pertenecer a un único core. Los puntos ruido no pertenecen a ningún clúster.

En este algoritmo tenemos menos margen de control con el número de grupos, ya que tan solo controlamos cantidades de puntos y distancias entre ellos. Esto hace que no podamos definir un número de grupos exactos, pero al mismo tiempo nos ayuda a explorar grupos más fuertes (con menor distancia y mayor requerimiento de puntos cercanos) y grupos más suaves (mayor distancia y menor requerimiento de puntos).



Diferencias entre k-means y DBSCAN. Fuente: DBSCAN. Github.com

DBSCAN tiende a tener mejores resultados que K-means cuando la complejidad de los datos es más grande. Pero esto puede ser un arma de doble filo, ya que complicar las relaciones puede llevar a grupos de datos que no tienen relación. Además, DBSCAN no puede seleccionar el número de grupos de forma directa, lo cual hace que K-means aporte más que DBSCAN.

- **Evaluación de modelos de clustering**

Como ya se ha mencionado, evaluar modelos de clustering no es fácil puesto que no tenemos unas *labels* que indiquen si los clústeres son correctos o no, pero sí podemos evaluar cómo de razonables son estos clústeres.

Una forma de evaluar los clústeres es con el **método Elbow**, el cual define cómo de compactos son los clústeres a través de un cálculo medio de las distancias entre puntos dentro de un clúster. Esta métrica tiende a beneficiar más a algoritmos de tipo K-means, puesto que los clústeres se definen cerca de un centroide.

Otra métrica muy usada dentro del análisis no supervisado es la de **silhouette**, la cual evalúa la distinción entre clústeres. Es una combinación entre separación y compresión de los clústeres. Esta métrica puede llegar a beneficiar a algoritmos como DBSCAN, pero puede ofrecer valores bajos si la separación entre clústeres es muy pequeña.

2

Reducción dimensional

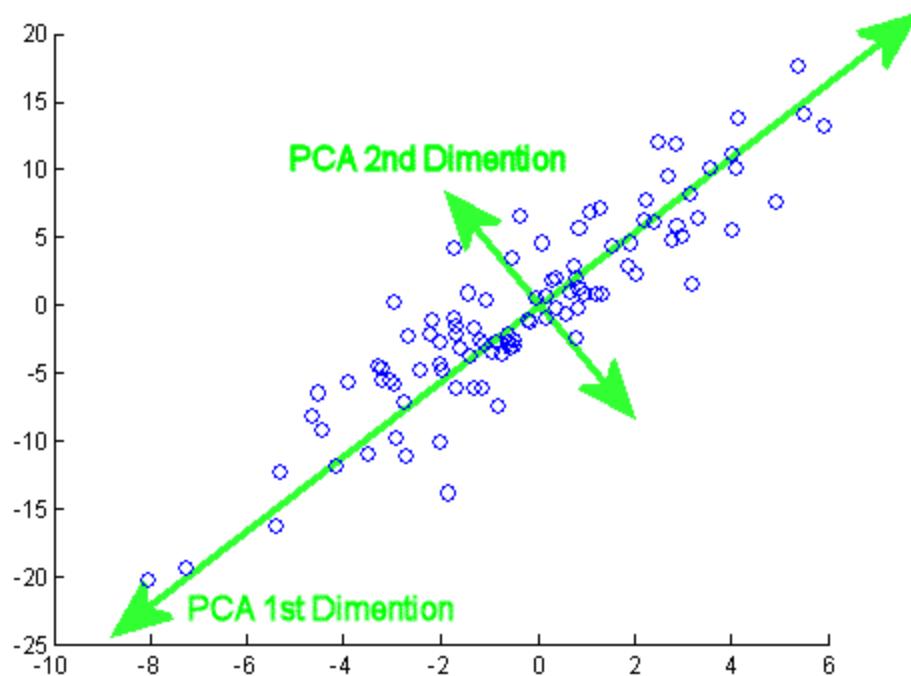
En múltiples casos tendremos la posibilidad de trabajar con miles de características de nuestros datos. Por un lado, esto es positivo, ya que nos permite tener una mayor granularidad de lo que hace cada dato único. Por otro, esto implica una mayor cantidad de ruido. Si fuésemos a entrenar un modelo de aprendizaje supervisado, podemos correr el riesgo de entrenar en un algoritmo que no sea capaz de generalizar.

Para estos casos tenemos los **métodos de reducción dimensional**. Estos modelos reducirán la cantidad de características con combinaciones entre ellas y, además, serán capaces de dictar qué características son más influyentes en nuestros datos.

- **Principal component analysis**

Principal Component Analysis (PCA) es un algoritmo que busca combinaciones de las características para encontrar la mayor variedad entre los datos, expresado en **componentes**. Esto permite reducir las dimensiones de nuestros datos, al encontrar las características que representan mayor variedad y al formar nuevos componentes.

Esto se consigue haciendo el cálculo de la covarianza de los datos (en grupos de dos, como un dato cambia con respecto a otro) para obtener los autovectores y autovalores, que serán llamados componentes. Los vectores nos indicarán qué valores representan mayor variabilidad y los valores cuánto es esa varianza en cada valor.



Aquí vemos un ejemplo de PCA como nuevos ejes. Fuente: *Principal component analysis (PCA): Explained and implemented*, de Raghavan. Medium.com

Los **vectores** se pueden usar como una nueva representación de los datos, pudiendo reducir datos de miles de características en dos componentes principales. Los **valores**, en cambio, permiten poder hacer un ranking de qué características distinguen mejor nuestros datos y por cuánto. Con esa información podríamos decidir quedarnos con unas características de los datos y descartar los que pudieran devolver más ruido, o podemos incluso usar los componentes como nuevas características.

- ***Linear discriminant analysis***

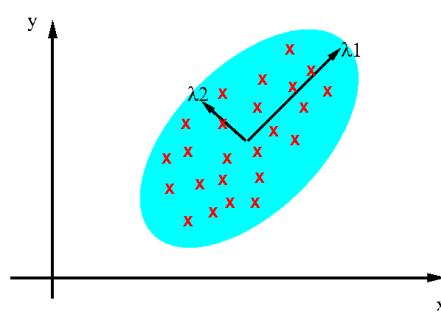
Linear Discriminant Analysis (LDA) tiene una funcionalidad similar a PCA pero, en vez de buscar las características que representan más variabilidad entre datos, representará las **características más distintivas entre clases**.

Para hacer esta separación necesitaremos tener clases. Pero insistimos, no vamos a aprender a clasificar los datos, sino a ver qué características son las que más distingue entre clases. Para ello, crearemos un eje que divida nuestras clases maximizando la distancia entre el valor medio de cada clase sobre este nuevo eje y que reduzca la separación de valores dentro de la misma clase.

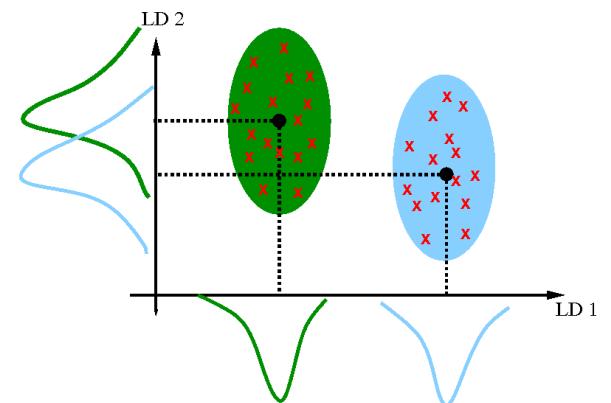
LDA genera un nuevo eje por cada dos clases, de tal forma que, si tenemos 3 clases, se necesitará dos ejes (creando un plano); 4 clases formarán un cubo, etc. Esto es un gran aporte del LDA, puesto que es más común tener un gran número de características que un gran número de clases. ¿Y qué significa esto?

Que si tenemos datos con miles de características que debemos dividir en 3 clases, podemos reducir el número de dimensiones a 2.

PCA: component axes that maximize the variance



LDA: maximizing the component axes for class-separation



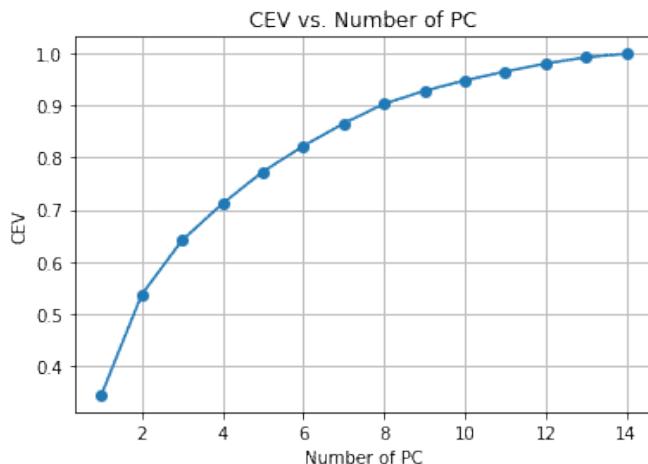
En estos gráficos podemos ver las diferencias entre PCA y LDA. Fuente:
<https://apsl.tech/en/blog/using-linear-discriminant-analysis-lda-data-explore-step-step/>

LDA es un algoritmo de reducción de dimensiones que busca distinguir entre clases.

Esto puede resultar útil para poder hacer una selección de características, previo a hacer una clasificación o como mejora de los resultados de un algoritmo. **Su limitación es la dependencia sobre esas clases.** Fuera de la clasificación, LDA no podrá hacer una distinción adecuada de los datos.

- **Evaluación de algoritmos de reducción dimensional**

Los algoritmos de reducción de dimensiones no tienen una evaluación por sí mismos, pero sí se pueden usar para hacer una **selección de características**. Una evaluación es su *Cumulative Expanded Variance (CEV)*, una gráfica la cual muestra cómo cada acumulación de componentes muestra la variación de los datos, es decir, nos permite hacer una selección de componentes para poder explicar un porcentaje de la variación de los datos.



Si observamos la curva CEV, podemos contemplar que el 90% de la variación de los datos se explica con los primeros 8 componentes. Fuente: *Exploring Unsupervised Learning Metrics*, de Cornelius Yudha Wijaya. Kdnuggets.com

Aunque esta métrica no juzgue al algoritmo, sí nos puede valer para entender la variabilidad de nuestros datos y qué influye sobre ellos.

Conclusiones



A modo de resumen, te presento aquí los bloques de contenido que hemos tratado en el fastbook:

- Los principales tipos de aprendizaje dentro de la inteligencia artificial y el machine learning: el supervisado y el no supervisado. Hemos visto las diferencias entre ambos aprendizajes y sus casos de uso.
- Bajo el marco de cada uno, hemos explorado los cuatro casos más comunes para el uso del machine learning: clasificación, regresión, clusterización y reducción de dimensiones.
- También hemos descubierto los distintos algoritmos que se usan para la resolución de los problemas. Esto nos ha llevado a una exploración detallada de varios algoritmos, lo que nos ha permitido aprender a utilizarlos.

De aquí en adelante, podremos resolver cualquier problema que corresponda a **datos tabulados**, ya sea para poder generar un modelo productivo como un análisis detallado de los datos.

Recursos útiles



Qualentum Lab

Para ampliar y reforzar los conocimientos adquiridos a través de esta lectura, recomendamos que visites los siguientes recursos:

- Vídeos dedicados para cada algoritmo que hemos visto:

<https://www.youtube.com/@statquest>

- La librería de Python con implementación de todos los algoritmos:

<https://scikit-learn.org/stable/>

- Una muy clara explicación sobre las diferencias entre los algoritmos de clustering:

<https://towardsdatascience.com/understanding-dbscan-algorithm-and-implementation-from-scratch-c256289479c5>

- Todos los detalles sobre métricas de aprendizaje no supervisado:

<https://www.kdnuggets.com/2023/04/exploring-unsupervised-learning-metrics.html>

¡Enhорabuena! Fastbook superado



Qualentum.com