OBS Business School

RED DE EDUCACIÓN SUPERIOR
Planeta Formación y Universidades

Raúl Melgosa García

Fundamentos de Machine Learning

Apuntes teóricos IA

Índice

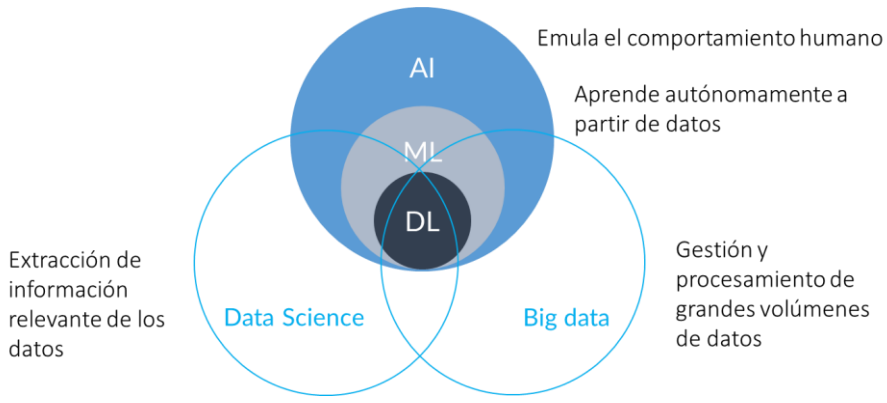
1. Introducción al Machine Learning

1.1. Qué es la Inteligencia Artificial y el Machine Learning

Una posible definición de Inteligencia Artificial es la capacidad de un sistema informático de comportarse de un modo semejante al de un humano inteligente en la resolución de un problema, capacidad que se alcanza a través del uso de algoritmos que se entrenan y ejecutan con datos.

Estos comportamientos "inteligentes" pueden ir desde la comprensión de imágenes, hasta la predicción de comportamientos o incluso, como hemos visto con el auge de los modelos LLM, la generación de texto con un cierto nivel de aparente creatividad. Las técnicas para lograr estos resultados son variadas, y con un nivel de complejidad diverso, pero una de las fundamentales es el **Machine Learning**, que abarca todos aquellos algoritmos capaces de construir, a partir de un conjunto de datos, modelos que pueden resolver un problema determinado, con la capacidad de reentrenarse (de forma automática o asistida) a partir de nuevos datos, mejorando así el comportamiento del modelo. Es esta última característica la que permite que estos sistemas tengan la apariencia de aprender por sí solos, mejorando sistemáticamente su rendimiento a medida que ganan experiencia.

A lo largo de este módulo conoceremos más de los sistemas de Machine Learning, sus principales tipos y usos, y cómo aplicarlos en el contexto empresarial. No es el objetivo de este módulo adentrarnos profundamente en los modelos matemáticos que hay detrás de estos algoritmos, ni aprender cómo programarlos en lenguajes como Python, sino entender sus fundamentos y aplicación.



1.2. El "boom" de la IA: contexto histórico y habilitadores

La Inteligencia Artificial ha estado presente en el imaginario colectivo desde hace décadas, permitiendo soñar con máquinas y sistemas con capacidades increíbles que han protagonizado todo tipo de relatos de ciencia ficción. Pero en los últimos años hemos comenzado a ver cómo

estos sistemas se iban acercando al presente, incluso en algunos casos convirtiéndose en realidad. Pero, ¿cuándo ha ocurrido este "boom"? ¿Qué es lo que lo ha propiciado?

La realidad es que la primera vez que se acuñó el término "Inteligencia Artificial" fue en 1956[1] por parte del informático John McCarthy, y algunos de los algoritmos que se emplean en la actualidad dentro de esta disciplina fueron desarrollados en 1980 (Kunihiko Fukushima). Entre esas primeras redes neuronales y el "boom" de la IA, que podemos situar alrededor del 2016, han pasado más de 35 años, ¿qué es lo que hay en la actualidad que no existiera en 1980 y ha permitido que una tecnología con interés teórico se convirtiera en una de las principales tendencias del mercado?

La respuesta se encuentra en dos ejes diferentes: **disponibilidad de datos** y **capacidad de computación**.

Respecto a la primera de estas causas, baste decir en este momento que en 2020 se generaron más de 30 veces más del volumen de datos que se crearon o replicaron en 2010[2] (datos de *statista*). Los datos, como veremos posteriormente, son el combustible que alimenta a los algoritmos de Inteligencia Artificial, y en particular a los de Machine Learning, a mayor volumen y variedad de datos, más posibilidades de crear modelos capaces de predecir el comportamiento humano, reconocer patrones u ofrecer algún tipo de respuesta inteligente. Citando a Eric Schmidt, antiguo CEO de Google, "entre los albores de la humanidad y 2003 se crearon 5 exabytes de información, mientras actualmente creamos ese mismo volumen cada dos días".

Pero no basta con acumular un volumen alto de datos para poder crear un sistema de Inteligencia Artificial, es necesario procesarlo de algún modo. Hasta hace unos años, la capacidad de procesamiento necesaria para poder trabajar con algoritmos de IA era inexistente, o excesivamente cara que, a efectos prácticos, implica lo mismo. Aquí es donde entra en juego un segundo factor que ha tenido un impacto de crecimiento exponencial en el desarrollo de la Inteligencia Artificial: **la nube**.

En el terreno de la IA, los modelos de servicio como el PaaS (Platform as a Service) o el IaaS (Infrastructure as a Service) que habilita la nube, nos permiten utilizar la potencia de grandes servidores, pagando sólo por el tiempo que la usemos. Así, podremos entrenar o ejecutar algoritmos complejos, usando capacidades avanzadas de Inteligencia Artificial sin necesidad de invertir en grandes máquinas que las procesen. La democratización de la capacidad de computación deriva directamente en la democratización del uso de la IA, puesto que se ofrece una alternativa para el entrenamiento de modelos que, de otra manera, sería impensable.

El aumento exponencial de los datos disponibles, el crecimiento de la capacidad de computación y el mayor acceso a ambas a través de la nube ha supuesto la tormenta perfecta para que en la actualidad se puedan explotar algoritmos que existen desde los años 80, permitiendo que den resultados mejores de forma inmediata.

[1] Más sobre los principales hitos en la historia de la IA en este artículo: https://www.nationalgeographic.com.es/ciencia/breve-historia-visual-inteligencia-artificial_14419

[2] Datos de Statista: https://es.statista.com/grafico/26031/volumen-estimado-de-datos-digitales-creados-o-replicados-en-todo-el-mundo/

1.3. Taxonomía de las tecnologías asociadas a IA

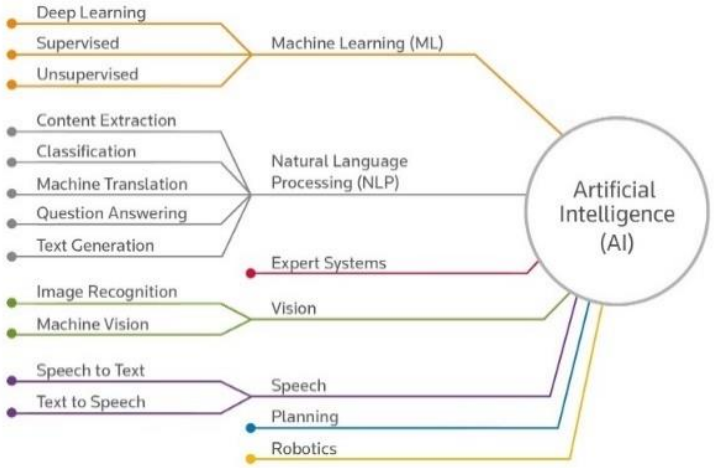
Existen muchas formas de categorizar las tecnologías asociadas a la Inteligencia Artificial. En primer lugar, nos centraremos en su complejidad. Si bien actualmente nos encontramos ante un posible punto de inflexión en las capacidades de la IA con sistemas como ChatGPT o GPT-4 a la cabeza, en los últimos años se podía establecer que la Inteligencia Artificial funcionaba de manera asombrosa en la ejecución de tareas muy concretas, pero no tan bien al abordar problemas más amplios. De esta manera, cuestiones aparentemente complejas, como la interpretación de imágenes o de lenguaje natural, se podrían considerar problemas suficientemente concretos como para obtener un buen rendimiento a través del uso de IA, mientras cuestiones más sofisticadas o abstractas, como el arte o el humor, estarían lejos de ser algo que una IA pueda resolver.

Desde este punto de vista, la primera categorización que podemos hacer de la Inteligencia Artificial separa los sistemas de propósito específico (Inteligencia Artificial Débil o ANI), de los sistemas generalistas (Inteligencia Artificial General o AGI) y de los sistemas capaces de superar las capacidades humanas (Superinteligencia o ASI). Hoy por hoy, cualquier sistema conocido es del primer tipo, **ANI**, si bien el avance en los agentes generalistas, a través de los LLM, nos hace pensar que la llegada de un sistema **AGI** podría estar a no más de una o dos décadas de distancia (la comunidad científica, hoy por hoy, no considera que este tipo de sistemas demuestre una inteligencia general, pese a ser capaces de resolver problemas de diversa índole, incluso sin haber sido específicamente entrenados para ello[3]). Por último, respecto a las **superinteligencias**, no existen hoy en día indicios racionales de que puedan llegar a crearse sistemas así, que, evidentemente, conllevarían graves problemas éticos: estamos hablando de sistemas capaces de superar al ser humano en cualquier disciplina, produciendo la singularidad tecnológica, como tantas veces se ha teorizado, con alarmantes resultados, en la ciencia ficción.

Por otra parte, podemos agrupar los sistemas de IA actuales (todos ellos dentro de la categoría de ANI) desde un punto de vista tecnológico (en función del tipo de algoritmos que utilizan) o funcional (en función de los problemas que resuelven). Bajo estas líneas, podéis encontrar una propuesta de taxonomía[4], que mezcla ligeramente ambos criterios, ya que los algoritmos de ML se utilizan en muchas de las demás categorías que se indican. Por ejemplo, los algoritmos de *computer vision*, *speech to text* y *text to* speech, se basan en redes neuronales, que pertenecen a la categoría de *Deep Learning*, dentro del *Machine Learning*.

[3] En este artículo de AI Time Journal se defiende este punto de vista: https://www.aitimejournal.com/gato-GPT-3-and-dall-e-what-are-generalist-agents-and-how-close-are-we-to-agi, postura en la que también se sitúa el matemático Noam Chomsky, como se puede leer en este artículo: https://culturainquieta.com/es/pensamiento/item/20093-la-critica-de-noam-chomsky-al-sistema-de-inteligencia-artificial-chat-gpt.html. También se pueden encontrar diversas opiniones que sitúan a GPT-3 o GPT-4 muy próximos al ámbito de la AGI, algunas se pueden leer en este artículo: https://towardsdatascience.com/GPT-3-a-complete-overview-190232eb25fd.

[4] Esta propuesta en particular se encuentra en https://www.automation.com/en-us/articles/august-2022/ai-machine-learning-human-intelligent-systems.



En este documento nos centraremos en las técnicas de ***Machine Learning*** y en sus aplicaciones para realizar predicciones con base en datos y para el aprendizaje por refuerzo de tareas complejas. Hablaremos primero de los algoritmos de aprendizaje supervisado, después de los algoritmos no supervisados, y finalmente introduciremos algunos algoritmos adicionales, como las series temporales, el aprendizaje por refuerzo y las redes neuronales.

1.4. Big Data: el dato como combustible del ML

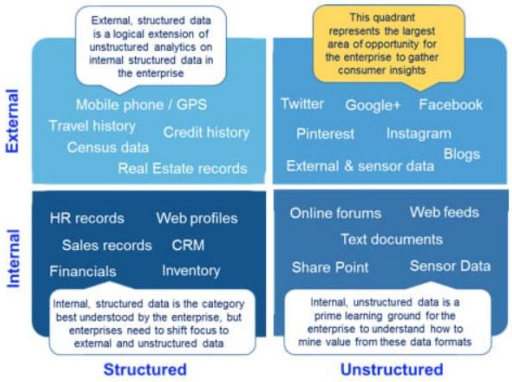
Como hemos comentado anteriormente, los sistemas de Inteligencia Artificial se alimentan de datos para aprender y poder realizar las funciones para las que son diseñados. En algunos casos, basta con algún centenar de registros para que un sistema pueda realizar con fiabilidad una tarea sencilla (clasificaciones en función de pocas variables), pero para casos más complicados, cuanto más, mejor. Y aquí "más" puede significar millones de registros.

El llamado Big Data representa estos volúmenes masivos de información, pero es mucho más que eso. La definición de Big Data se establece a través de las llamadas "tres uves":

- **Volumen**: para hablar de Big Data, por supuesto, tenemos que referirnos a un alto volumen de información, en términos de número de ficheros y espacio de almacenamiento, que puede llegar a superar el Terabyte[5].
- **Variedad**: uso de distintas fuentes de datos y distintos formatos procesados de forma conjunta. Esto implica tanto datos estructurados (tablas, ficheros XML), como datos no estructurados (fotografías, documentos legales, grabaciones de sonido).
- **Velocidad**: la tercera característica del Big Data es la velocidad de procesamiento. En muchos de los problemas de analítica actuales, las respuestas tienen que ser inmediatas (*real time*).

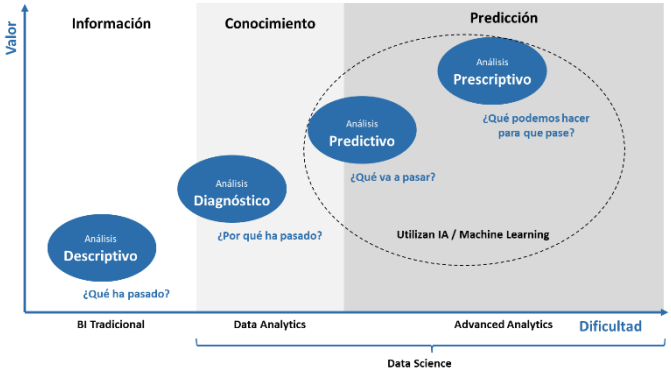
[5] Un Terabyte son 1000 Gigabytes, que a su vez son 1000 Megabytes. Por encima del TB se encuentran los Petabytes (1000 TB), los Exabytes (1000 PB) y los Zetabytes (ZB). El volumen de información que se estima que se creó o replicó en 2020 alcanza los 64 Zetabytes.

En función de su estructura, los datos pueden ser **estructurados** (tienen una estructura definida y fija, que puede ser interpretada a través de reglas simples) o **no estructurados** (como las imágenes o el lenguaje natural, que no pueden ser interpretados de forma directa por una máquina). Además, al hablar de Big Data podemos considerar tanto datos **internos** y **externos**, según su procedencia. En el esquema del lateral[6] se muestran algunos ejemplos de cada tipo.



El uso de datos para la toma de decisiones de negocio es anterior al boom de la IA, se desarrolla fundamentalmente bajo la disciplina conocida como *Business Intelligence*. Se trata del conjunto de métodos y herramientas que permiten extraer conocimiento (*insights*) a partir del análisis de datos estructurados e internos de una organización. Los datos se convierten en información, la información en conocimiento y este conocimiento se pone en marcha para tomar decisiones que permitan entregar valor al cliente u optimizar la operativa del negocio. Podemos distinguir cuatro tipos de analítica diferentes que, ordenados en función de su complejidad y del valor aportado serían:

- **Análisis descriptivo**: permite saber qué es lo que ha pasado en una situación concreta (cómo se han distribuido los beneficios entre los distintos productos o por geografías).
- **Análisis diagnóstico**: permite entender por qué ha pasado algo (qué variable ha tenido más incidencia en la caída de cartera, cuál es el perfil de cliente que ha dado más beneficios, etc.).
- **Análisis predictivo**: permite anticipar qué es lo que va a ocurrir en un escenario en particular (cómo van a evolucionar las contrataciones de producto o la cartera de clientes ante ciertas circunstancias).
- **Análisis prescriptivo**: permite identificar qué acciones son las más indicadas para propiciar que ocurra un escenario buscado (por ejemplo, qué producto ofrecer a un determinado cliente).



A medida que utilizamos técnicas de análisis más complejas, el trabajo humano necesario para tomar una decisión se simplifica, hasta llegar al análisis prescriptivo, que es capaz de decidir de forma autónoma. Estas técnicas de *Advanced Anlytics* que permiten llegar a ese punto utilizan tecnologías de Inteligencia Artificial, principalmente, Machine Learning.

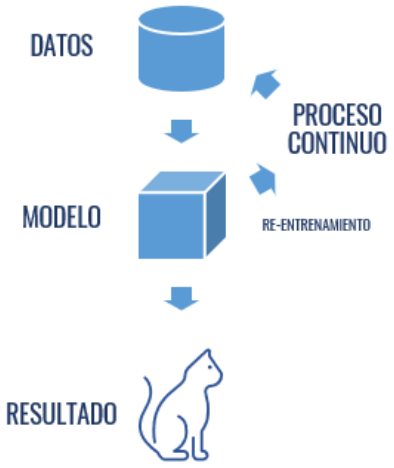
[6] Fuente del gráfico: Cambridge Strategic Management, http://csm-cambridge.com/services/data/types-of-data

1.5. Machine Learning: conceptos básicos y tipos

Llamamos *Machine Learning* (ML), o **algoritmos de aprendizaje automático**, a un conjunto de algoritmos que permiten mejoran su rendimiento ("aprenden") a medida que se nutren de nueva información. La explosión de la IA tal y como la conocemos hoy, coincide con la explosión del uso del ML. Ya en 2016, Google se definió como una compañía "*Machine Learning first*"[7], expresando así que su prioridad era la de embeber capacidades de aprendizaje automático en todos sus productos.

Para entender en qué consiste exactamente ML, empezaremos por definir qué es un **algoritmo**: se trata de un conjunto ordenado y finito de operaciones que permite hallar la solución de un problema. Los métodos con los que se aprende a hacer una multiplicación o una división a mano en el colegio son algoritmos. En el mundo de la informática, desde la invención de los primeros "antepasados" de los ordenadores modernos, la forma en la que se le indica a un sistema cómo tiene que actuar ante unas entradas determinadas de datos es a través de algoritmos. De hecho, los primeros algoritmos pensados para su ejecución por parte de una máquina datan del siglo XIX, los escribió (o programó) **Ada Lovelace** en tarjetas perforadas para su ejecución en la máquina analítica de Babbage[8]. Desde entonces las máquinas han seguido procesando información siguiendo la misma idea de fondo: contar con reglas inequívocas y ordenadas que les permiten tomar decisiones.

Así, en la computación tradicional, los algoritmos constituyen una serie de reglas que permiten llegar a una conclusión en función de los valores de entrada. Estas reglas pueden ser más o menos complicadas, contar con más o menos nodos de decisión, pero siempre son unívocas y explicables, siempre podemos llegar a saber por qué la máquina ha proporcionado una salida determinada. Sin embargo, ML supone un cambio de paradigma: en los sistemas de ML los algoritmos se utilizan para crear un modelo complejo, que se nutre de un volumen masivo de datos y que posteriormente podrá tomar decisiones que no siempre sabremos explicar. A medida que se obtengan nuevos datos, el algoritmo podrá generar mejores modelos en un proceso llamado "re-entrenamiento", que serán capaces de dar un resultado mejor.



A modo de ejemplo, si queremos crear un modelo de ML que sea capaz de identificar imágenes de gatos, alimentaremos un algoritmo de reconocimiento de imágenes con una serie de imágenes

[7] Más detalles en este artículo de Wired: https://www.wired.com/2016/06/how-google-is-remaking-itself-as-a-machine-learning-first-company/
[8] Más sobre Ada Lovelace y la máquina analítica de Babbage en este artículo del NY Times: https://www.nytimes.com/es/2018/03/10/espanol/cultura/ada-lovelace-obituario-overlooked.html

categorizadas como “gato” o “no gato”. Ese algoritmo generará un modelo que, ante una nueva imagen sin categorizar, nos dirá, con un intervalo de confianza, si esa imagen es la de un gato o no. A medida que recojamos un mayor volumen de imágenes de gatos, podremos volver a entrenar el mismo algoritmo con más datos, de manera que su rendimiento mejore.

Repasaremos ahora algunas de las principales características de los algoritmos de ML:

- **Dependen de los datos**: cuantos más y mejores datos tengamos, mejores serán los modelos creados por estos algoritmos. Es importante cuidar que la calidad del dato sea la adecuada, no sólo en veracidad, sino también en formato. Por ejemplo, datos de naturaleza continua, como fechas, pueden necesitar un preprocesamiento para dar una información relevante: si hablamos de la fecha de nacimiento de un grupo de clientes, puede que este dato no tenga tanto valor como el rango de edad que conseguiríamos agrupando los registros con fecha de nacimiento entre dos valores determinados.
- **Producen sistemas deterministas, pero de caja negra**: los modelos generados por algoritmos de ML siempre devolverán los mismos resultados ante los mismos datos de entrada (es decir, son deterministas), sin embargo, no siempre seremos capaces de entender por qué toman las decisiones que toman. Sí podemos, en algunos casos, saber qué variables de entrada han tenido un mayor impacto en la toma de decisiones, lo cual puede dar una pista suficiente. En ocasiones, puede ser conveniente trabajar de forma simultánea con varios algoritmos: uno que da los resultados más precisos, pero sin devolver información alguna sobre las decisiones tomadas, y otro, con resultados menos precisos, pero que informa sobre las variables que más peso tienen en la decisión.
- **Trabajan con intervalos de confianza**: en el ejemplo anterior, el modelo que reconoce imágenes de gatos no daría como respuesta un “gato” o “no gato”, sino el porcentaje de probabilidad de que la imagen esté mostrando un gato. Dependerá de la lógica de negocio que estemos empleando que decidamos a partir de qué porcentaje para nosotros es suficiente para considerar el resultado como “gato” – es decir, dependiendo de cómo de negativo sea cometer un error, exigiremos una mayor certeza por parte del algoritmo para dar su respuesta como válida. Hablaremos más de esto y de las matrices de confusión en el bloque 4, “*Proyectos de IA*”.
- **Pueden ser reentrenados**: como comentábamos anteriormente, a medida que obtengamos nuevos datos, podemos reentrenar el algoritmo de ML para generar un nuevo modelo, más eficaz. Se produce así un círculo virtuoso en el que, cuanto mejor es un algoritmo, más se utiliza, y cuanto más se utiliza, más datos recoge para su reentrenamiento, permitiendo de este modo un reentrenamiento que lo haga aún mejor. Este proceso de reentrenamiento se puede automatizar, sin embargo, puede llegar un momento en el que, usando el mismo algoritmo, no conseguiremos una mejoría significativa al ingerir nuevos datos, y habrá que modificar la parametrización del algoritmo, cosa que, hoy por hoy, no es tan fácilmente automatizable.

Existen multitud de algoritmos de ML, con distintos niveles de complejidad y distintos ámbitos de aplicación. Los podemos agrupar en dos grandes categorías en función de cómo utilizan los datos para generar sus modelos: algoritmos supervisados y no supervisados. En los siguientes capítulos hablaremos de cada uno de estos grupos, dejando para el cuarto otros algoritmos relevantes.

2. Algoritmos Supervisados

2.1. Qué son los algoritmos supervisados

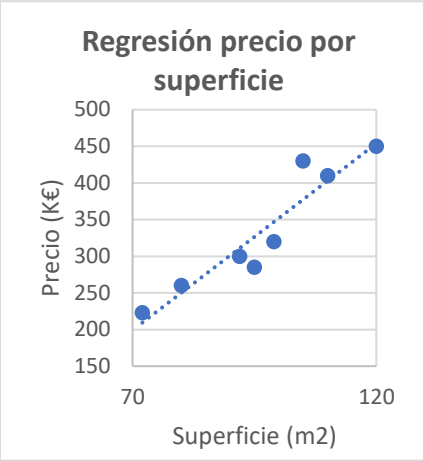
Los **algoritmos supervisados** son aquellos en los que el set de datos que se utiliza para entrenar el modelo contiene también la variable objetivo que se pretende obtener como respuesta de este. Es decir, en el ejemplo del gato que utilizábamos anteriormente, el algoritmo que utilizamos sería supervisado, ya que el set de datos para el entrenamiento contiene imágenes categorizadas, en las que se indica si se trata de un gato o no.

Son algoritmos que se utilizan para la realización de predicciones sobre una variable objetivo en función de otra serie de variables. Se entrenan con un set de datos que contiene una serie de registros (cuantos más, mejor) con todas las variables que se puedan incluir, incluyendo la variable objetivo. En el ejemplo del gato, sólo tenemos dos variables: la propia imagen y la variable objetivo (gato o no gato), pero lo más habitual es que los registros contengan múltiples variables y, generalmente, cuantas más, mejor.

Imaginemos que queremos construir un modelo que prediga si un cliente de un banco va a incurrir en el impago de un crédito. Para alimentar nuestro modelo, construiríamos un set de datos con toda la información histórica que pudiéramos recopilar de los clientes que ha tenido el banco. En este set, cada registro correspondería a un cliente, utilizaríamos variables como la edad del cliente, su antigüedad en el banco, si cuenta con otros productos financieros, su dirección, la oficina a la que pertenece, si tiene o no domiciliada su nómina (y, en caso de tenerla, la cuantía de sus ingresos mensuales), etc. Cuanto más sepamos del cliente, mejor trabajará el modelo. A esos registros le incluiremos una última variable: si cada uno de esos clientes ha incurrido en un impago o no. Con esto, un algoritmo supervisado podría devolver un modelo que, ante un registro que contenga toda la información anterior, excepto la variable objetivo, nos devolvería la probabilidad de que el impago se produzca.

La gran potencia de los algoritmos de ML radica en que son capaces de identificar patrones donde a primera vista no los hay. Siguiendo el ejemplo anterior, con un enfoque de reglas de negocio, podríamos tratar de llegar al mismo resultado, estableciendo que los clientes en una cierta franja de edad, por debajo de unos ciertos ingresos y sin otros productos vinculados al banco tienen una mayor probabilidad de impago. Y puede que esto funcione relativamente bien, pero estaríamos obviando mucha más información que está a nuestra disposición y que no podemos procesar. Es habitual que un algoritmo supervisado se entrene utilizando cientos de miles de registros y que cada uno de esos registros contenga decenas de variables. El ser humano, sin apoyo de la tecnología, no es capaz de identificar patrones en conjuntos de datos de esta magnitud.

2.2. Regresiones lineales y logísticas



Dentro de los algoritmos supervisados, a su vez, podemos encontrar algunos relativamente sencillos y otros de gran complejidad. Los más sencillos, de hecho, se llevan utilizando mucho tiempo a nivel profesional, por ejemplo, en el mundo financiero y asegurador se utilizan para predecir la rentabilidad de los clientes y establecer estrategias de *pricing*. Se trata de las **regresiones lineales**. En su expresión más sencilla, la regresión lineal simple es una técnica que permite identificar la línea que mejor se adapta a un conjunto de puntos. En el ejemplo adjunto se puede observar una gráfica que describe la relación del precio de la vivienda en un barrio determinado, con los metros cuadrados de superficie de cada inmueble (los datos son meramente ilustrativos). En esta gráfica, cada punto representa una vivienda de la que conocemos únicamente estos dos valores. La regresión lineal simple trata de encontrar cuál es la línea que está más cerca de pasar por todos estos puntos, es decir, que minimiza la suma de las distancias que hay entre cada uno de los puntos y ella misma. Para conseguir dar con esa línea, se utilizan expresiones matemáticas que, en este caso, no presentan excesiva complejidad. Sin embargo, ¿qué ocurriría si quisiéramos incluir en el modelo información sobre la antigüedad de la vivienda o el número de cuartos de baño? La propia representación se va complicando un poco, ya que tendríamos que utilizar gráficos en tres dimensiones, o mucho si optamos por incluir las dos variables, pero la matemática subyacente sigue pudiéndose aplicar, solo que ahora es más compleja. Si el número de variables se eleva drásticamente, la dificultad para ejecutar el algoritmo y generar un modelo, también se dispara.

Una regresión lineal trata de encontrar la función $y = \beta_0 + \beta_1 x + \varepsilon$ que mejor se ajuste al juego de datos de la muestra, para ello es necesario encontrar el valor de los parámetros β_0 y β_1. La variable ε representa error, que refleja la variabilidad de la función al margen de la relación entre la variable dependiente y la independiente. Para encontrar los parámetros β_0 y β_1 se puede utilizar el método de mínimos cuadrados, consistente en encontrar los valores que minimizan la suma de los cuadrados de las desviaciones entre los valores observados de la variable dependiente (es decir, los valores reales de y de los que se dispone al entrenar el modelo) y los valores de y que se obtendrían con la fórmula anterior. Así, para cada punto de la muestra tendremos un valor y_i real y un valor y'_i obtenido a través de unos parámetros β_0 y β_1, y el objetivo del método de mínimos cuadrados es minimizar la función $\sum(y_i - y'_i)^2$. Desarrollando la fórmula llegamos a:

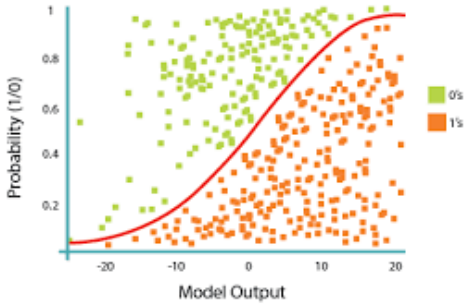
$$\beta_1 = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - (\sum x_i)^2}$$

$$\beta_0 = \frac{\sum y_i - \beta_1 \sum x_i}{n}$$

Donde n es el tamaño de la muestra. La misma lógica matemática se puede extrapolar a un conjunto mayor de variables independientes, complicando significativamente el cálculo de los parámetros, que pasa a realizarse de forma matricial. Por supuesto, lenguajes de programación

especializados en *machine learning* cuentan con librerías que facilitan la implementación de estos algoritmos (y del resto que veremos en estos apuntes), no obstante, el foco de esta asignatura no es profundizar ni en desarrollo matemático, ni en la programación de estos métodos, sino explicar sus fundamentos y aplicación profesional.

Por otra parte, los algoritmos de **regresión logística** también parten de una formulación matemática que trata de identificar la curva que mejor separa dos juegos de datos. En este caso, son algoritmos que se utilizan para clasificación, es decir, pueden identificar si el valor de una variable es "verdadero" o "falso", no devuelven un valor concreto, como sí hace la regresión lineal.



La formulación matemática de una regresión logística busca encontrar la curva que mejor separa los distintos valores de la muestra de datos, esta curva se basa en la función sigmoidea:

$$f(x) = \frac{1}{1 + e^{-x}}$$

El valor devuelto por la función logística resultante se encuentra entre un rango de 0 a 1, la respuesta se redondea para obtener una respuesta booleana que sirva para realizar una clasificación de verdadero (1) o falso (0). También se puede adaptar una regresión logística para realizar clasificaciones multinomiales, es decir, para muestras de datos con más de dos categorías que tengan una relación matemática entre ellos, por ejemplo, establecer cuatro categorías de viviendas en función de su probabilidad de aumentar su precio, que pueda ser muy baja (<25%), baja (<50%), alta (<75%) y muy alta (>75%). En este caso, el resultado de la regresión logística multinomial se interpreta en esos cuatro segmentos.

Igual que en el caso anterior, incorporar nuevas variables independientes al problema complica la expresión matemática del mismo, pero la lógica detrás de esta formulación es la misma.

2.3. Árboles de decisión

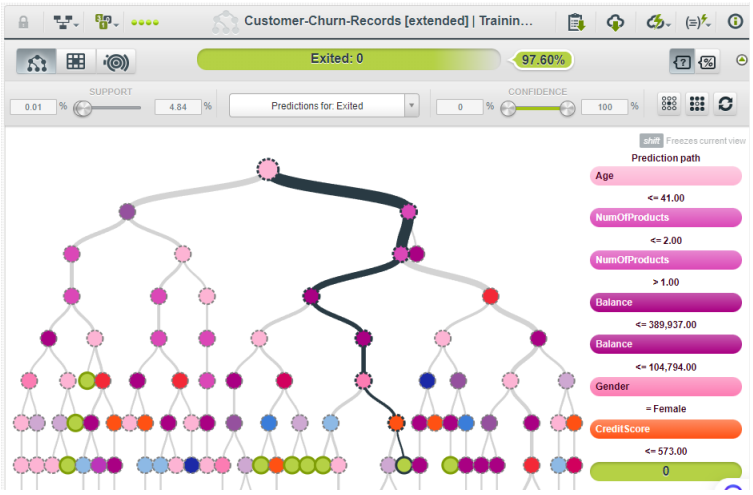
Existen más tipos de algoritmos supervisados más complejos que las regresiones lineales y logísticas y que pueden dar mejores rendimientos en problemas complicados. Los **árboles de decisión** son algoritmos que crean, de una forma automática, una sucesión encadenada de nodos conectados en una estructura arbórea. Cada nodo representa una variable y cada rama un rango de posibles valores que puede tomar, hasta llegar a unos nodos finales que indican el valor que ha de tomar la variable objetivo con un cierto grado de confianza, que variará en función de la ruta por la que se haya llegado.

Existen distintos algoritmos que permiten generar un árbol de decisión a partir de un dataset. Se trata, en cualquier caso, de algoritmos recursivos que establecen un nodo eligiendo un atributo y

su valor de corte con los que segmentar el juego de datos. A continuación, se aplican de forma iterativa sobre los juegos de datos segmentados resultantes hasta cumplir con una condición de finalización.

La forma de seleccionar los atributos y el valor de corte varía entre los distintos algoritmos, incluso en un mismo algoritmo se pueden usar distintos métodos. Por ejemplo, el algoritmo ID3 (Iterative Dichotomister 3), define una función de entropía que permite establecer el atributo que proporciona una mayor ganancia de información, a partir de dicho atributo, se divide el dataset en dos, en función del "voto mayoritario" de la muestra, si la variable es categórica, o del valor promedio del atributo en los ejemplos de la muestra, si la variable es continua. A continuación, la misma lógica se vuelve a aplicar en cada subconjunto de la muestra.

Los árboles de decisión se validan evaluando en cada nodo el atributo correspondiente, y siguiendo al nodo adecuado, en función de la respuesta. En el ejemplo siguiente, construido en BigML, se puede apreciar resaltado una hipotética evaluación de un registro de entrada en un árbol de decisión, en el que vemos que para un cliente de al menos 41 años, con al menos 2 productos contratados, un balance inferior o igual a 104794€, mujer y con *credit score* menor o igual a 573, la probabilidad de que permanezca en la compañía es del 97,6%.

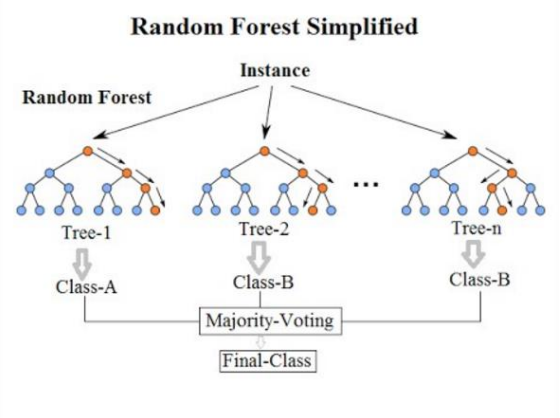


Cabe destacar que los árboles de decisión pueden considerarse tanto para problemas de clasificación como para problemas de regresión.

2.4. Random Forests

Cuando entrenamos un árbol de decisión, estamos configurando una lógica que depende de los valores presentes en el data set de entrenamiento. Si dentro de ese juego de datos seleccionáramos un subconjunto, excluyendo aleatoriamente parte de los valores de la muestra, el árbol de decisión resultante sería diferente. Así, podemos generar distintos subconjuntos de la muestra aleatorios y que cada uno genere su propio árbol. Ésta es la lógica que se encuentra detrás del siguiente algoritmo que vamos a analizar, el ***random forest*** (bosque aleatorio). En esta técnica el resultado final del modelo es el valor más repetido a lo largo de los distintos árboles que lo

componen (en problemas de clasificación), o bien la media de los valores retornados por cada uno (en problemas de regresión), tal y como se aprecia en la imagen adjunta[9].



A la hora de confeccionar cada árbol de decisión dentro de un random forest, se pueden también seleccionar algunos de los atributos disponibles, de manera independiente y aleatoria para cada árbol, consiguiendo reducir la correlación en los resultados producidos por cada uno de ellos. Esta técnica se denomina **"feature bagging"**.

Los principales beneficios del uso de random forest frente a un árbol de decisión sencillo son los siguientes[10]:

- **Se reduce el riesgo de "overfitting"**: el problema del "overfitting" o sobreajuste deriva del sobreentrenamiento de un algoritmo que produce que sus resultados sean muy precisos para el juego de datos de entrenamiento, pero no generalicen bien para otros datos. Aunque cada árbol que compone un *random forest* pueda sobreajustar su data set parcial, la combinación de todos ellos tenderá a neutralizar este problema de sobreajuste.
- **Flexibilidad**: los *random forests* son, como decíamos, algoritmos adecuados para problemas tanto de regresión como de clasificación, resolviéndolos con un grado de precisión más que aceptable. Además, el *feature bagging* permite reducir el impacto producido por características faltantes en un data set.
- **Permiten identificar la importancia de los atributos**: con estos algoritmos es sencillo evaluar qué variables han tenido más peso en la decisión del modelo, algo que se puede conseguir con distintas técnicas, como MDI (disminución media de la impureza) que determina cuánto peor se comporta un modelo al excluir cada variable.

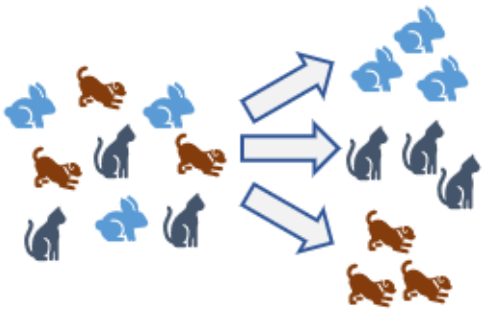
No obstante, los random forest son algoritmos sensiblemente complejos, que requieren un largo tiempo y recursos tanto para su entrenamiento como para su ejecución, con lo que pueden no ser la solución más indicada para problemas sencillos, en los que otros modelos puedan aportar una solución razonablemente precisa.

[9] Imagen obtenida de Wikipedia: https://en.wikipedia.org/wiki/Random_forest
10 Más sobre random forest en este enlace de IBM: https://www.ibm.com/topics/random-forest

3. Algoritmos No Supervisados

3.1. Qué son los algoritmos no supervisados

El siguiente grupo de algoritmos que vamos a analizar son los **algoritmos no supervisados**, que se caracterizan por no utilizar en su entrenamiento ninguna variable objetivo, sino que identifican patrones en el juego de datos a través de los que pueden crear distintas subcategorías. Volviendo al ejemplo del reconocimiento de imágenes de gatos que poníamos anteriormente, un enfoque no supervisado de este problema consistiría en entrenar al modelo con imágenes de distintos animales, por ejemplo, gatos, perros y conejos, sin acompañar estas imágenes de ninguna variable adicional, es decir, sin decirle al algoritmo qué es lo que está viendo. El algoritmo entrenaría al modelo para que pudiera identificar que existen tres grupos distintos de imágenes, A, B y C, sin saber qué representa cada uno, y para poder categorizar una nueva imagen en uno de esos grupos.



Los algoritmos no supervisados[11] se utilizan para la clasificación y agrupación de variables. Un ejemplo de aplicación sería la identificación automática de perfiles de clientes, a partir de la información que dispongamos de ellos en nuestro CRM. Las agrupaciones jerárquicas, *K-means*, *isolation forests* y algunas redes neuronales (de las que hablaremos más adelante) son ejemplos de algoritmos de ML no supervisados. En general, podemos hablar de dos grandes grupos de algoritmos no supervisados: los algoritmos de clasificación y los de detección de anomalías. A continuación, entraremos en detalle en estos dos tipos.

3.2. Algoritmos de clasificación

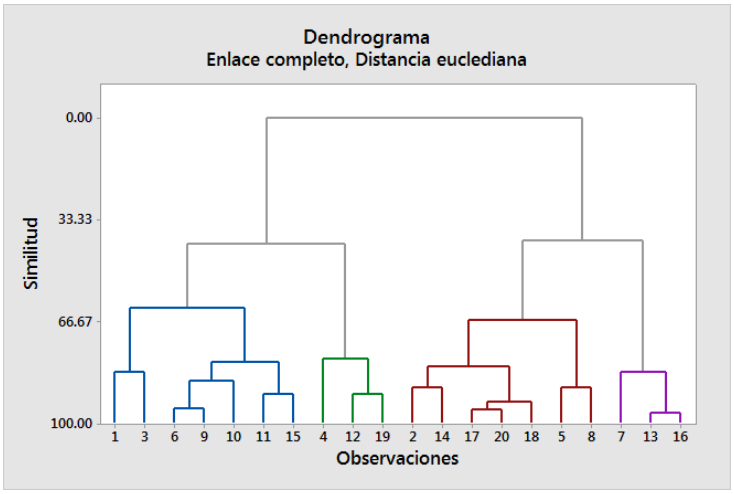
Este tipo de algoritmos tratan de encontrar patrones comunes en el juego de datos de entrenamientos que permitan diferencias distintas categorías en los datos. Hablaremos de algunos de estos algoritmos.

Para empezar, los **algoritmos de agrupación jerárquica** de clústeres buscan construir un árbol que organice los datos de la muestra según su distancia. Cada nodo de este árbol representa un subconjunto de datos con un cierto grado de similitud. Cuando más cerca estamos de los nodos hojas, más similitud hay en el clúster recogido por un nodo, mientras cuanto más nos alejamos, más vaga es la relación. Existen varias técnicas de agrupación jerárquica, según el árbol se construya desde las hojas hasta la raíz (tipo aglomerativo) o desde el nodo inicial hasta las hojas

[11] Algún artículo sobre aprendizaje no supervisado en estos enlaces: https://www.tibco.com/es/reference-center/what-is-unsupervised-learning y https://www.alexanderthamm.com/es/blog/asi-funciona-el-aprendizaje-maquina-sin-supervision/

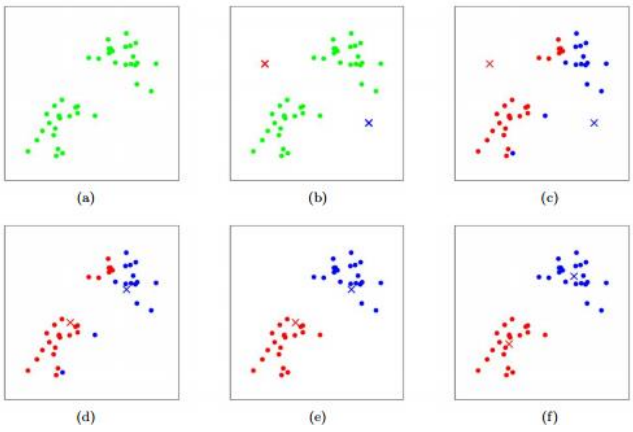
(divisibles), y también en función de cómo se mida la distancia entre los clústeres, que puede ser consideran sus puntos más alejados, sus puntos más cercanos, la distancia entre las medias de los puntos y la distancia promedio entre pares.

Estos algoritmos presentan la ventaja de que no necesitan que se defina previamente el número de clústeres a clasificar, sino que este se deriva de forma directa de la aplicación del modelo. Sin embargo, no funcionan bien con datasets muy grandes y su resultado puede variar sensiblemente en función del método utilizado para medir las distancias entre clústeres. Sus resultados se visualizan en dendogramas que representan el grado de similitud que se observa en los clústeres, decreciente según ascendemos desde las hojas del árbol hasta el nodo raíz. Junto a estas líneas se puede observar un ejemplo de un dendograma en el que se recogen hasta seis jerarquías de clasificación dentro del árbol (en las observaciones 6 y 9 y en 17 y 20). Aceptando como razonable un nivel de similitud del 66,67%, el dendograma del ejemplo nos sugeriría estableces un total de cuatro categorías diferentes, maracas en distintos colores. Si buscamos un nivel de similitud mayor, el número de clústeres aumenta.



Otro ejemplo de algoritmo no supervisado de clasificación es **K-means**, un algoritmo de clusterización basado en **centroides**, esto significa que la forma en la que el algoritmo identifica si un dato pertenece a una categoría o a otra es su distancia al centroide correspondiente para cada categoría. K-means se ejecuta eligiendo un cierto número (k) de centroides iniciales de manera aleatoria (en el paso b de la imagen inferior, vendrían representados por las marcas roja y azul, pero podríamos tener más centroides si se buscara establecer una clasificación en más de dos categorías). A continuación, se categoriza cada punto de la muestra en función de su proximidad a cada centroide: si están más cerca del centroide "rojo", son puntos rojos, y viceversa (se ve en el paso c de la imagen).

Como podemos ver en este ejemplo, esto ha separado la muestra completamente en dos categorías, pero la separación no responde a ningún criterio razonable: el siguiente paso será ajustar cómo elegimos los centroides para segregar mejor la muestra. Para ello, desplazaremos cada centroide en la dirección de las medias de los puntos que pertenecen a su clúster. Es decir, calcularemos la media (vectorial) de todos los puntos rojos y



moveremos el centroide rojo en esa dirección y sentido, y haremos lo mismo para el centroide azul. El paso (d) muestra este desplazamiento de centroides, mientras el (e) muestra cómo se vuelve a evaluar cada punto de la muestra respecto a los nuevos centroides, consiguiendo una categorización que refleja mucho mejor la realidad.

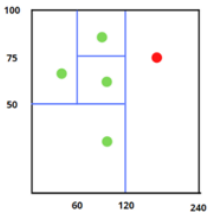
En este ejemplo, ejecutar una única vez el algoritmo ha sido suficiente para conseguir centroides que segregan la muestra de un modo razonable: los puntos rojos y azules separan el juego de datos en dos áreas que ya se podían identificar como diferentes en la imagen (a). No obstante, la idea de este algoritmo es continuar su ejecución iterativa, volviendo a elegir centroides cada vez. El algoritmo detendrá su ejecución al alcanzar un criterio de convergencia establecido (que las distancias medias de los puntos a su centroide sean inferiores a un cierto umbral, que los centroides no cambien o que ningún punto cambie de clúster) o tras un número de ejecuciones determinada, sin importar el grado de convergencia alcanzado. Para evaluar la clasificación de un nuevo dato a través del modelo generado por K-means bastará con identificar cuál es su centroide más próximo.

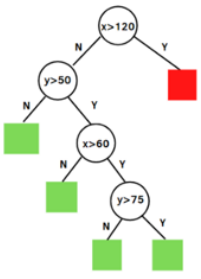
K-means presenta algunas desventajas, como la necesidad de establecer previamente el número de clústeres a identificar en la muestra, o la asunción de que los patrones presentes en el conjunto responden a una geometría esférica. También es muy sensible a valores atípicos y tiende a confeccionar clústeres de tamaño homogéneo: si en nuestro juego de datos hay un clúster pequeño y compacto, frente a otro más amplio y disperso, la respuesta de K-means tenderá a ser incorrecta.

3.3. Algoritmos de detección de anomalías

Algunos algoritmos de aprendizaje no supervisado tienen como objetivo identificar valores extraños dentro de un conjunto de datos[12], que pueden representar fraudes, incidentes, errores o cualquier otro tipo de anomalías. Son algoritmos indicados para identificar excepciones en un juego de datos, y esto es una virtud relevante, ya que los algoritmos supervisados no suelen generalizar bien cuando existen pocos ejemplos del valor objetivo en la muestra. A continuación, veremos un par de estos algoritmos.

Los ***isolation forests***, son árboles de decisión específicamente diseñados para detectar anomalías en un conjunto de datos. Su configuración es bastante intuitiva. Igual que otros algoritmos vistos hasta ahora, se generan de forma recursiva. En cada iteración, la muestra se segrega a partir de un atributo escogido al azar, dividiéndola en dos en función de un valor, también aleatorio, de dicho atributo. Este procedimiento se repite





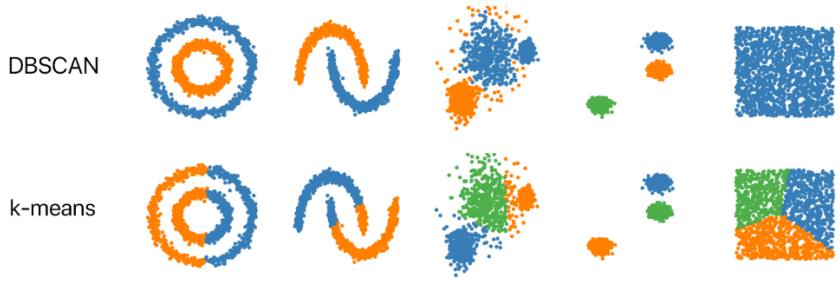
[12] Más sobre estos algoritmos: https://centum.com/aprendizaje-no-supervisado-para-la-deteccion-de-anomalias/

recursivamente hasta que estas segregaciones dejan un valor aislado, o bien a todos los valores del mismo lado, creando un árbol de decisión que aísla (*isolate*) los valores anómalos.

La principal ventaja de los *isolation forests* frente a otros algoritmos no supervisados similares es que requieren una limitada capacidad de procesamiento, lo que los hace adecuados para trabajar con grandes volúmenes de datos con muchas dimensiones. Sin embargo, no son algoritmos fáciles de interpretar (caja negra) y son bastante sensibles al ruido del dataset de entrenamiento[13].

Por otra parte, otros algoritmos como **DBSCAN** (Density-Based Spacial Clusterion of Applications with Noise) están específicamente indicados para la detección de ruido en un dataset. El DBSCAN sigue una lógica de algún modo similar a K-means[14], trata de identificar puntos centrales que agrupen los datos de la muestra en clústeres según sus puntos vecinos, considerando como anomalía (ruido) todo valor que quede "fuera" de esos clústeres. Es decir, a diferencia de K-means, en DBSCAN no se asigna un clúster a cada punto, sino que se delimita el "área de influencia" de los centroides, estableciendo vecindarios en la muestra y considerando como anomalía todo valor que no caiga dentro de uno de estos vecindarios.

Otra diferencia relevante entre K-means y DBSCAN es que este segundo establece los vecindarios de cada punto central a partir de la densidad de valores cercana, siendo capaz de configurar clústeres con distintas propiedades geométricas, no necesariamente esféricos, como se aprecia en la figura inferior[15]. Esto implica que es un algoritmo que también puede comportarse bien en ciertos problemas de clasificación.



[13] Más sobre Isolation Forests, específicamente sobre su implementación en Python, en este enlace: https://anderfernandez.com/blog/isolation-forest-en-python/

[14] Para más información sobre la comparación de ambos algoritmos podéis consultar esta guía: https://pieriantraining.com/dbscan-vs-kmeans-a-guide-in-python/

[15] Fuente: https://towardsdatascience.com/understanding-dbscan-algorithm-and-implementation-from-scratch-c256289479c5

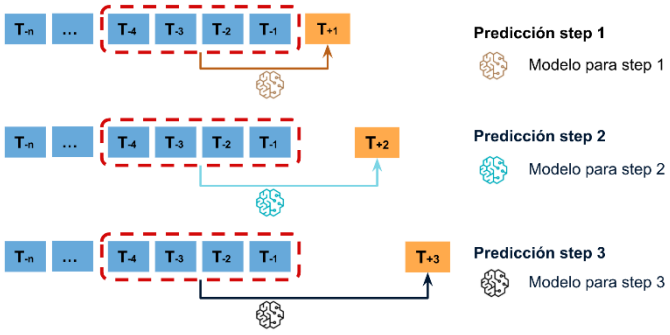
4. Series Temporales y otros algoritmos

4.1. Series temporales

Muchos de los problemas que se pretenden resolver con técnicas de machine learning consisten en predecir el comportamiento de una cierta variable a lo largo del tiempo. Las series temporales buscan abordar este tipo de problemas, trabajando sobre datasets conformados por registros tomados en intervalos regulares de tiempo.

Este tipo de registros pueden mostrar comportamientos regulares o con una cierta variación estacionaria (por ejemplo, las ventas de productos pueden mostrar picos en períodos de rebajas, o el precio de vuelos y hoteles puede aumentar en temporadas altas de turismo. El acercamiento más sencillo a las series temporales es identificar el comportamiento de la serie y asumir que la estacionalidad existente se va a repetir en el futuro. Es decir, si durante los últimos años se ha identificado que en noviembre tenemos un crecimiento de ventas promedio del 50% (asociado al Black Friday), podremos suponer que las ventas del próximo noviembre serán un 50% superiores a las observadas en octubre.

La observación de series temporales permite afinar el comportamiento de otros modelos predictivos, a través de técnicas que se aplican sobre el entrenamiento de los mismos, como el **pronóstico de ventana móvil**, consistente en incorporar, por orden, cada valor del set de pruebas al set de entrenamiento, produciendo un nuevo modelo con el que predecir el siguiente valor. Cabe destacar que, en estos casos, el set de pruebas no debe elegirse de forma aleatoria, sino que ha de consistir en las últimas muestras de la variable presentes en el dataset. Esta forma de evaluar un modelo con series temporales es más realista, ya que refleja el comportamiento que podríamos tener en el mundo real, donde el modelo podría ser reentrenado periódicamente con las últimas predicciones realizadas. La ventana de entrenamiento puede ser fija (usando siempre el mismo número de valores y descartando los más antiguos cada vez, es la representada en el gráfico adjunto[16]) o incremental (considerando siempre toda la muestra), en función de si los datos cambian más o menos a lo largo del tiempo.



En cuanto a qué algoritmos son más útiles para la predicción de series temporales, cabe destacar **ARIMA** (AutoRegressive Integrated Moving Average), capaces de aislar el comportamiento estacional de la serie de datos de su ruido, lo que mejora su capacidad de predecir. El término "autoregresivo" indica que cada valor predicho en la serie se relaciona con los valores pasados de la misma variable, mientras que "media móvil" se refiere a que el error de regresión se calcula por

[16] Fuente: https://cienciadedatos.net/documentos/py27-forecasting-series-temporales-python-scikitlearn.html

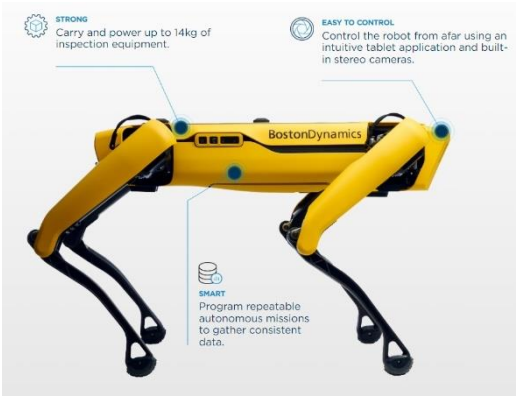
la combinación lineal de errores pasados. Finalmente, el término "integrado" hace referencia al proceso por el cual se busca forzar la estacionariedad del algoritmo, consistente en considerar no cada observación de la variable en el tiempo, sino su diferencia con las observaciones previas.

Otra forma de abordar series temporales es usando redes neuronales, de las que hablaremos un poco más adelante. En términos generales, se puede concluir que ARIMA es una mejor opción cuando la correlación temporal de los datos es alta, mientras las redes neuronales suelen ser más fáciles de aplicar y poner en producción.

4.2. Algoritmos de aprendizaje por refuerzo

Hemos visto numerosos algoritmos capaces de hacer predicciones y clasificaciones, en algunos casos partiendo de datos etiquetados (supervisados), otras veces identificando de forma automática patrones en el juego de datos (no supervisados), y finalmente especializados en desentrañar la estacionalidad de una serie de datos repartidos en intervalos fijos de tiempo (series temporales). El **aprendizaje por refuerzo** da un paso más allá, creando sistemas que mejoran su rendimiento, de forma automática, cada vez que se ejecutan, siendo capaces de reconocer qué comportamientos deben replicar en el futuro y cuáles deben evitar.

Los sistemas de aprendizaje por refuerzo se utilizan para la realización de tareas complejas y están diseñados para identificar cuándo su respuesta es correcta o errónea. Después de cada ejecución de la tarea, el modelo vuelve a entrenarse, incorporando la información recogida de su última iteración; es decir, incorporando como un dato más de su entrenamiento el hecho de que una cierta actuación fue positiva (lo que reforzará la repetición de este comportamiento) o negativa (lo que derivará en que ese comportamiento se evite en el futuro).



Spot, el popular perro robot de la compañía Boston Dynamics, es un ejemplo de sistema entrenado con aprendizaje por refuerzo. El sistema que permite que este robot camine en distintas superficies no se rige a través de reglas fijas que interpretan qué hacer exactamente en cada posible circunstancia que se encuentre, sino que es un modelo capaz de identificar cuándo el robot ha llegado a su objetivo y cuándo se ha caído o bloqueado, y que incorpora la información de cada desplazamiento en su reentrenamiento.

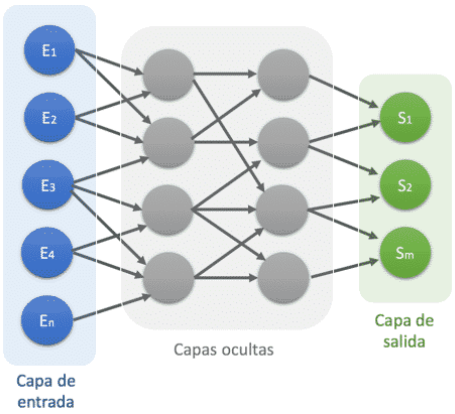
Otro ejemplo que ilustra la potencia de estos algoritmos es el juego del escondite creado por Open AI[17]. Se trata exactamente de lo que parece: agentes virtuales que juegan al escondite. Unos agentes tienen como objetivo ver a los otros, mientras los segundos tienen que intentar no ser vistos. Estos objetivos son los que condicionan su reentrenamiento. Lo fascinante de este ejemplo es cómo a lo largo de las iteraciones, los agentes virtuales no sólo aprenden a esconderse, sino

[17] Más sobre este caso en la web de OpenAI: https://openai.com/blog/emergent-tool-use/

que también empiezan a utilizar herramientas de su entorno para mejorar su rendimiento, sin haber sido entrenados específicamente para ello.

4.3. Introducción a redes neuronales

Al margen de la categorización establecida de los distintos tipos de algoritmos de ML, cabe mencionar las **redes neuronales**, una subfamilia de algoritmos, dentro de la cual existen casos supervisados, no supervisados y de refuerzo. Las redes neuronales buscan emular de forma artificial el comportamiento del cerebro humano. Igual que nuestro cerebro está formado por un entramado de neuronas interconectadas por sinapsis, las redes neuronales artificiales son un entramado de nodos, organizados en una serie de capas, que se conectan entre sí a través de sinapsis con un cierto peso numérico.



La capa de entrada describe, mediante una serie de variables codificadas en los valores de los nodos que la configuran, la situación ante la que la red neuronal debe dar una respuesta. Cada valor de estos nodos se multiplica por el peso de la sinapsis correspondiente al trasladarse al siguiente. En las capas ocultas, se realiza una operación con los valores precedentes (concretamente, se suman todos esos productos de los valores de la capa anterior por los pesos de las sinapsis de unión, y posteriormente se aplica una función de activación que normaliza el resultado) y se vuelve a propagar el resultado a la siguiente capa, una vez más, multiplicándolo por el peso de la siguiente sinapsis. Finalmente, los valores de los nodos de la capa de salida describen la respuesta que tiene que dar el sistema ante el valor de la capa de entrada.

La red neuronal más sencilla posible, una red monocapa o perceptrón simple, que consta únicamente de una capa oculta. En este caso, la capa oculta procesa las entradas de la red multiplicando cada una por el peso sináptico correspondiente, sumando los resultados y aplicando la función de activación que los normaliza para llegar al resultado final. Al añadir más capas ocultas a este modelo obtenemos un perceptrón multicapa, cuyo funcionamiento es igual, pero las salidas de cada nodo de la primera capa oculta se tratan como entradas para la siguiente, volviendo a multiplicarse por los pesos sintácticos, sumarse y aplicarse a dicha suma la función de activación. De esta manera, en cada nodo de una capa oculta se está produciendo la siguiente operación:

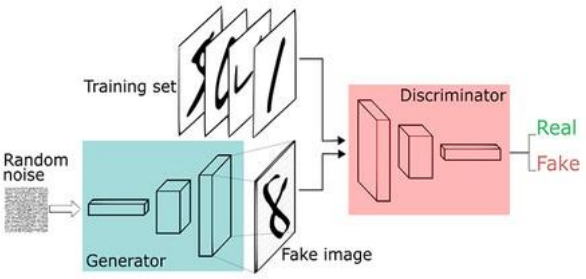
$$y_i = f(\sum w_{ij} x_j - \theta_i)$$

Donde y_i representa la salida del nodo "i" dentro de la capa "y", $f(x)$ es la función de activación, x_j es el valor procedente del nodo "j" de la capa anterior ("x"), w_{ij} es el peso sináptico de la unión entre los nodos "i" y "j" de las dos capas en cuestión y θ_i es un valor umbral que ayuda a normalizar las salidas.

Los algoritmos de entrenamiento de una red neuronal establecen qué peso deben tener las distintas sinapsis para adaptarse a los valores del set de datos de entrenamiento. Cuando la red se reentrena, no cambia el número de capas ocultas, ni las funciones de activación, ni el número de sinapsis presentes, solamente el peso de las sinapsis. Como decíamos inicialmente, este modelo se asemeja a la forma en la que nuestro cerebro procesa la información y, por difícil de comprender que pueda resultar, funciona.

Las redes neuronales se utilizan para diversas funciones de clasificación, predicción y agrupación, constituyen el llamado "deep learning". El reconocimiento de imágenes o de voz, son ejemplos de tareas que se resuelven con esta tecnología. También se emplean en los algoritmos de aprendizaje por refuerzo que mencionábamos anteriormente.

Un ejemplo particular de red neuronal, que está especialmente en auge en la actualidad, son las **redes generativas adversarias** (GAN)[18]. Se trata, una vez más, de dos sistemas que trabajan de forma coordinada, con el objetivo último de crear un contenido falso. Uno de los sistemas actúa como generador, y está diseñado para producir tal contenido, mientras el segundo actúa como discriminador, y está diseñado para identificar cuándo un contenido es real y cuándo es falso. Cada uno de los sistemas utiliza la respuesta del otro para reentrenarse, a través de aprendizaje por refuerzo, y perfeccionar cada vez más su rendimiento. A medida que el discriminador se vuelva más eficaz identificando contenido falso, el generador se verá forzado a producir un contenido que parezca más real, y viceversa. El funcionamiento conjunto de ambos sistemas deriva en un círculo virtuoso que permite la creación de contenidos que, hoy por hoy, pueden engañar a un humano.



Si bien las GAN pueden tener aplicaciones interesantes en el mundo del marketing y la creación de contenidos, también conllevan importantes implicaciones éticas, al poder tener un uso claramente pernicioso y difícil de detectar. De estas implicaciones hablaremos más adelante, en el apartado dedicado a la ética de la IA.

Como decíamos al comienzo de este documento, las técnicas de Inteligencia Artificial tienen un gran rendimiento para resolver problemas específicos, pero son más ineficaces ante problemas más amplios. Sin embargo, a través del trabajo conjunto de distintos algoritmos diseñados para resolver pequeñas partes de un problema, el resultado que podemos obtener es asombroso.

[18] Más información aquí: https://www.xataka.com/inteligencia-artificial/conceptos-inteligencia-artificial-que-gans-redes-generativas-antagonicas. Fuente original del gráfico, Skymind.

5. Proyectos de ML y herramientas

5.1. Gestión de un proyecto de ML: fases y metodologías

El objetivo de esta sección es centrarnos en las particularidades de la gestión de proyectos de Machine Learning, no en profundizar en metodologías de gestión de proyectos. Pero, ¿qué tienen de particular los proyectos de ML para requerir una gestión especializada? La respuesta más directa es que son proyectos con un alto grado de incertidumbre, de carácter innovador y disruptivo. Es importante abordarlos con metodologías ágiles con el objetivo de poder reducir riesgos, y estar listos para pivotar y reenfocar al proyecto si se encuentra alguna barrera insalvable.

Podemos definir cinco grandes fases dentro de su ejecución. Es importante entender que no todas tienen que ocurrir de manera completamente secuencial (en particular, el desarrollo del modelo y la integración en los procesos de negocio son tareas que se pueden paralelizar hasta un cierto punto). Podríamos incluir una fase 0 consistente en la creación del equipo de proyecto, pero dejaremos la reflexión sobre los roles necesarios en una iniciativa de IA para la siguiente sección. A continuación, hablaremos de cada una de estas fases en más detalle.

Identificación del problema

En primer lugar, debemos tener claro cuál es el problema que queremos resolver. En esta fase se pueden aplicar técnicas de *Design Thinking*[19] o *Lean Start-up* para identificar con claridad dónde podemos estar aportando un mayor valor al cliente, pero lo fundamental es que el resultado de este ejercicio incluya un entendimiento de dónde está el valor de nuestro proyecto (cuál es el *business case* que lo sostiene), qué le aporta al cliente y cómo vamos a medir si tiene éxito.

Identificación y preprocesamiento de las fuentes de datos

Una vez tenemos claro el objetivo de nuestro proyecto, pasaremos a identificar todos los datos con posible relevancia que puedan ser útiles en el modelo. Es importante tener en cuenta que estos datos cumplan las siguientes características:

1. **Veracidad** – es evidente que no podemos construir un modelo sobre datos que no sean fiables, ya que las conclusiones podrían ser incorrectas.
2. **Accesibilidad** – una vez entrenemos nuestro modelo, a la hora de ejecutarlo necesitaremos disponer de todos los campos del registro para el que lo queramos ejecutar. Por ejemplo,

[19] En este artículo de *nexocode* se profundiza en la aplicación de *Design Thinking* en proyecto de IA: https://nexocode.com/blog/posts/applying-design-thinking-to-ai/

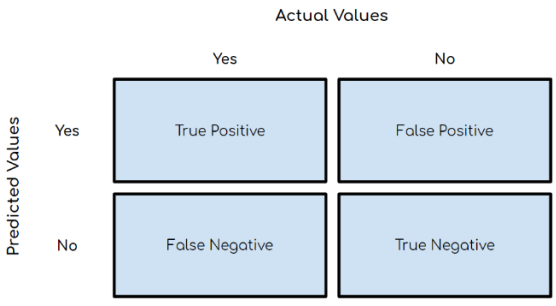
si hemos usado fuentes de datos externas para completar nuestro set de entrenamiento, tenemos que estar seguros de que en el momento de ejecutarlo vamos a tener acceso online a dichas fuentes.

3. **Relevancia** – es posible que a priori no sepamos seguro qué datos serán relevantes y cuáles no. Ante la duda, es preferible usar todos los datos disponibles y después descartar aquellos con menor peso en las decisiones del modelo, cosa que descubriremos en una fase posterior del proyecto.
4. **Trazabilidad** – cuando se utilizan campos procedentes de distintos orígenes de datos, tenemos que asegurarnos de que los podremos cruzar adecuadamente para componer un único *dataset* en el que cada registro tenga la información combinada de estos orígenes de datos.
5. **Representatividad** – a la hora de construir un juego de datos de entrenamiento, es importante que éste esté equilibrado, sobre todo en categorías potencialmente sensibles, como el sexo, la edad o categoría racial. Hablaremos más sobre ello en el último apartado de este tema.

Una vez identificados los datos que vamos a utilizar, los preprocesaremos para facilitar el entrenamiento de los modelos. El preprocesamiento incluye tareas como la limpieza de los datos, la normalización, clusterización o agregación, etcétera. Gran parte del éxito de un proyecto de ML depende de estas tareas[20]. Normalmente, existirán ciertas iteraciones entre este paso y el siguiente, incluyendo o quitando campos, o revisando cómo se explotan en función de los resultados proporcionados por el modelo.

Desarrollo del modelo

Normalmente, no entrenaremos un único modelo, sino que utilizaremos distintos algoritmos para entrenar varios modelos y ver cuál funciona mejor. Para ello, en primer lugar, tendremos que dividir nuestro juego de datos de entrenamiento en dos: un 80% de los registros se utilizarán efectivamente para entrenar el modelo (conteniendo, si se trata de un algoritmo supervisado, el valor de la variable objetivo en cada registro) y el 20% restante compondrá nuestro set de datos de prueba, en el que inicialmente ocultaremos el valor de la variable objetivo. Una vez tengamos nuestros modelos entrenados compararemos su respuesta con los datos del set de prueba.



Para ello se pueden utilizar varias técnicas, como las matrices de confusión, consistentes en representar gráficamente cuánto ha acertado y fallado el modelo, distinguiendo los falsos positivos de los falsos negativos. En la página anterior podéis ver el aspecto que tiene esta matriz. El objetivo del modelo será maximizar el número de predicciones acertadas (es decir, que los verdaderos positivos y los verdaderos negativos cubran aproximadamente el 100% de los datos

[20] Más sobre la limpieza y transformación de los datos en este artículo de quantdare: https://quantdare.com/data-cleansing-and-transformation/

del set de pruebas). Sin embargo, esto normalmente no ocurrirá. Para determinar si un modelo es mejor que otro, tenemos que valorar no sólo cuál acierta más, sino también cuál falla "mejor".

Dicho de otro modo, dependiendo de la lógica de negocio con la que integremos el modelo en nuestros procesos, un falso negativo y un falso positivo no tendrán el mismo coste para la empresa. Imaginemos que se trata de un modelo que busca identificar a clientes potencialmente buenos para ofrecerles automáticamente un crédito preconcedido. En este caso, un falso negativo implicará que el modelo no ofrezca el crédito a un cliente que en realidad era apto para ello – el coste es una potencial pérdida de negocio. Por otro lado, un falso positivo implicará que el modelo ofrezca el crédito a un cliente que en realidad no es apto para ello – el coste es un potencial riesgo de impago. Dependerá de la estrategia de negocio (si buscamos un crecimiento agresivo o si preferimos ser conservadores con los riesgos) el determinar qué tipo de fallo es mejor, pero lo que es seguro es que son fallos diferentes.

Otro factor relevante a la hora de desarrollar el modelo, y para el que también se utilizarán herramientas como la matriz de confusión es establecer la línea que separa el positivo del negativo. Recordemos que los modelos de *machine learning* no trabajan con certezas, sino con intervalos de confianza que podemos expresar como un porcentaje. El modelo anterior realmente no dice si un cliente es apto o no es apto para un crédito, sino que da el porcentaje de confianza que tiene de que lo sea. Nosotros decidimos a partir de qué porcentaje consideramos que el cliente es apto y esto volverá a depender de la estrategia de negocio y de nuestra tolerancia a los errores que pueda cometer el sistema. También podemos interpretar los resultados del sistema con tres posibles categorías: apto, no apto y "revisar", de manera que reduzcamos aún más el riesgo de error.

Integración en los procesos de negocio

Asumiendo que se cuenta con los datos adecuados, con las herramientas actuales y trabajando con un equipo de *data scientists* experto, construir un modelo de ML que dé buenos resultados es relativamente rápido. Ponerlo en producción es otra historia, sobre todo si se trabaja en un entorno informático con sistemas antiguos (sistemas *legacy*). Se puede empezar a trabajar en esta integración de manera paralela a la construcción del modelo, empezando por aquellos componentes que se tengan completamente claros – esta es una de las razones por las que un enfoque *agile* para estos proyectos es más que recomendable.

Por ejemplo, en el caso anterior, tenemos claro que la respuesta del modelo será "apto" o "no apto", y que el sistema informático tendrá que ejecutar la preconcesión del préstamo para los clientes que sean aptos. Se puede empezar a trabajar en esta parte mientras el modelo se desarrolla.

Sin embargo, lo que no sabremos hasta más adelante es qué variables vamos a acabar utilizando (recordemos que existirá una iteración entre la identificación de los datos y el desarrollo del modelo, y que se podrá descartar el uso de algunos campos si se percibe que no aporta valor al resultado final). Parte de las tareas de integración implican habilitar que esos datos estén disponibles a la hora de ejecutar el modelo, cosa que no siempre será inmediata. Por ejemplo, si uno de los datos que hemos utilizado para entrenar el modelo es el *credit scoring* del cliente, que

nos proporcionó el equipo de Marketing y Clientes en una Excel, tendremos que encontrar la forma de que ese mismo dato se pueda pasar al modelo en tiempo real cuando se vaya a ejecutar (típicamente, esto puede requerir la creación de servicios web que faciliten los datos que no están disponibles dentro del entorno analítico donde se ejecuta el modelo).

Monitorización y reentrenamiento

Por último, una vez que nuestro modelo esté creado, probado e integrado en los procesos de negocio, empezaremos a medir su rendimiento. En la primera fase del proyecto nos aseguramos de definir cómo lo haríamos, ahora es el momento de ponerlo en marcha. La particularidad de los proyectos de ML en cuanto a monitorización es que ésta impacta de forma directa y automática en la mejora continua del proceso: en el momento en el que el rendimiento de nuestro modelo baje de un cierto umbral que definamos, éste se volverá a entrenar con nuevos datos, con el objetivo de mejorar.

Pero, ¿de dónde va a sacar esos nuevos datos? Depende del proceso concreto en el que estemos trabajando. El caso más claro es el de modelos que se está utilizan para automatizar una actividad que anteriormente era manual. Lo que haremos en estos casos es mantener un volumen de operación manual de forma constante en el tiempo. Por ejemplo, una de cada diez operaciones que hay que realizar, las seguirá haciendo una persona, y almacenaremos el resultado de esas operaciones junto con el set de entrenamiento previo de nuestro modelo. Cuando el rendimiento del modelo caiga por debajo del umbral que establezcamos, lo volveremos a entrenar usando este set de datos ampliado, enriquecido con los datos de las nuevas operaciones realizadas manualmente.

Es posible que llegue un momento en el que esto no sea suficiente, y el rendimiento del modelo se mantenga bajo por más que lo reentrenemos con nuevos registros. En esos casos, toca iterar, el equipo de *data scientists* tendrá que entender qué está ocurriendo, si hay que cambiar de algoritmo, eliminar ciertos campos del modelo o incorporar otros.

5.2. Implicaciones éticas y legales del ML

Como a menudo ocurre con la irrupción masiva de tecnologías tan disruptivas, existen numerosos riesgos asociados al uso malintencionado de la Inteligencia Artificial, pero en este caso, los efectos negativos de la IA también pueden ser involuntarios, como ya ha ocurrido en alguna ocasión en los últimos años. Es por ello por lo que es importante conocer estos riesgos y mantener una perspectiva ética en el diseño y ejecución de las soluciones de IA. A continuación, repasaremos algunos ejemplos especialmente notorios de estos riesgos y algunos factores críticos a tener en cuenta.

En 2016, Microsoft lanzó al mundo un *bot* conversacional, Tay, diseñado para interaccionar por Twitter con usuarios de entre 16 y 24 años. El objetivo del *bot* era realimentar su entrenamiento con las interacciones que mantuviera, desarrollando así una personalidad sobre la marcha. En tan solo un día de actividad, pasó de decir que los humanos eran “super guay” a decir que era una

buena persona, pero solamente "odiaba a todo el mundo", además de exhibir comportamientos racistas y xenófobos[21]. Microsoft pasó a desactivar el *bot*.

Un año antes, Google había pasado por una situación similar con su aplicación Fotos, en la que se etiquetaba a personas racializadas como gorilas. Cuando el problema saltó a la prensa[22], la reacción de Google fue vetar la búsqueda de gorilas.

En ambos casos, no había ningún tipo de mala intención por parte de los dos gigantes tecnológicos a la hora de desarrollar estos algoritmos, sin embargo, los datos con los que se entrenaron ambos fueron causantes de este funcionamiento indeseado. En el caso de Tay, el problema fue no contar con ningún tipo de control sobre el juego de datos, no había ningún límite a la información que Tay podía recibir, y toda ella se utilizaba en su reentrenamiento. El *bot* fue "atacado" con comentarios racistas que incorporó en su modelo, acabando por reflejarlos. El caso de Google es algo más complicado, ya que Google sí tenía mayor control sobre el set de datos de entrenamiento de su sistema. El problema es que el balanceo incorrecto de la muestra hacía que Google Fotos no tuviera ningún problema en reconocer a personas de origen caucásico, mucho más presentes en su set de entrenamiento; mientras que las personas de color, con una presencia menor en la muestra, suponían una dificultad mayor para el modelo.

Un problema completamente distinto es el que representan los sistemas que se pueden utilizar con fines perjudiciales, principalmente en cuanto a la generación de documentos falsos. Pueden ser casos relativamente leves, como usar GPTChat para hacer pasar por propio un artículo o un trabajo, o casos graves, como generar imágenes falsas que se puedan utilizar como pruebas falsas o que puedan dañar la reputación de una persona. Precisamente por este riesgo, algunas herramientas como DALL-E están diseñadas con algunos límites[23], como el de no producir imágenes con contenido sexual o violento.

Por último, existen algunos dilemas éticos que no tienen una solución evidente. Todos los casos que hemos comentado son claramente sancionables, pero si hablamos de cómo se tiene que comportar un coche autónomo en caso de que un mal funcionamiento implique un accidente inevitable, si debe priorizar salvar a los pasajeros del coche o a un implicado externo, o si esta priorización debe depender de algún otro factor, como el número de vidas en peligro. Una prueba de que no todos estaremos de acuerdo en la resolución de estos debates es el ejercicio de crowdsourcing "The moral machine"[24], desarrollado por el MIT, en colaboración con otras universidades. Se trata de una página que presenta una serie de escenarios de accidentes inevitables en los que cada usuario debe elegir cuál es el menos malo. Después de contestar una serie de dilemas, la página muestra los resultados del usuario (qué ha priorizado en función de sus respuestas) y los compara con los recogidos en toda la comunidad.

[21] Se puede leer algo más de esta historia en este artículo de la BBC: https://www.bbc.com/news/technology-35890188
[22] De nuevo, más información en este artículo de la BBC: https://www.bbc.com/news/technology-33347866
[23] En este artículo de Genbeta se explora qué es capaz de hacer DALL-E y qué límites tiene: https://www.genbeta.com/a-fondo/dall-e-2-guia-a-fondo-que-como-funciona-todo-que-necesitas-saber-ia-generacion-imagenes-openai
[24] Podéis echar un vistazo a este ejercicio en su web: https://www.moralmachine.net/

Sin embargo, volviendo a los dos primeros casos, sí hay una serie de principios que debemos tener en cuenta para evitar que nuestros modelos de ML desarrollen comportamientos indeseados, asociados a la incorporación de sesgos en sus juegos de datos que afecten a **categorías protegidas**, como puede ser la edad, el sexo, la religión, o el color de piel:

- **Evitar, directamente, el uso de las categorías protegidas**, es decir, a la hora de entrenar un algoritmo, no utilizar datos que puedan considerarse sensibles. Un modelo no puede desarrollar un sesgo por edad, si no integra la variable edad o el sexo (ni ninguna otra con una correlación directa).
- **Garantizar una distribución equilibrada en la muestra de entrenamiento**, es decir, que en la muestra que se emplea para entrenar el algoritmo, se encuentren volúmenes similares de casos de cada categoría protegida, incluso aunque esa categoría no se pretenda explotar de forma directa. Esto habría evitado el caso de los gorilas de Google que comentábamos anteriormente.
- **Considerar a los grupos sensibles o protegidos en el momento de la calibración del modelo**, buscando la paridad estadística en los mismos en el resultado final. Esto es, poner especial foco en estas cuestiones a la hora de verificar los resultados de un modelo, y no dar por bueno un sistema de IA que sugiere distintos resultados sobre una población de hombres frente a una de mujeres, por ejemplo. Fallar en este aspecto podría conducir a que los sistemas aprendieran y perpetuaran sesgos actualmente presentes en la sociedad, tal y como se identificó en 2021 que ocurría con la publicidad de Facebook[25], donde anuncios de ciertas ofertas de empleo, tienen más opciones de mostrarse a hombres que a mujeres, y viceversa.

Conociendo los riesgos asociados al uso inapropiado de la inteligencia artificial, es fácil entender que una regulación específica al respecto es necesaria. Algunos de los problemas y desigualdades que pueden producirse ante la presencia de sesgos en estos algoritmos son suficientemente graves como para que no baste con depender de la buena fe de sus desarrolladores a la hora de entrenarlos, siguiendo los principios vistos anteriormente.

La Unión Europea ya se ha pronunciado al respecto[26], elaborando una propuesta de regulación compuesta por tres iniciativas:

- Un **marco legal** europeo que establezca los derechos fundamentales y los riesgos de seguridad específicos de los sistemas de IA.
- Un **marco de responsabilidad civil** adaptando las normativas de responsabilidad a la era digital y a la IA.
- Una revisión de las **legislaciones de seguridad sectoriales** en línea con los dos anteriores.

[25] Más sobre este caso en este enlace: https://www.wsj.com/articles/facebook-shows-men-and-women-different-job-ads-study-finds-11617969600

[26] Se puede encontrar este marco regulatorio en este enlace: https://digital-strategy.ec.europa.eu/en/policies/regulatory-framework-ai, y un resumen del mismo, por la firma de abogados Allen & Overy, aquí: https://www.allenovery.com/en-gb/global/news-and-insights/publications/key-provisions-of-the-draft-ai-regulation

El marco legal está orientado a riesgos, sin asumir que cualquier sistema de IA debe ser tratado de la misma forma, ya que no tienen las mismas aplicaciones. Se define un primer nivel de riesgo inaceptable, que implicaría la prohibición directa del desarrollo de ciertos sistemas, como son aquellos que puedan entrañar peligros físicos o psicológicos para personas, que definan parámetros de evaluación personal aplicados por autoridades públicas o que apliquen métodos de identificación biométrica en espacios públicos.

En el siguiente nivel se encontrarían los sistemas de alto riesgo, entre los que se encuentran aplicaciones de muy diversa índole, desde la asignación de tareas en algunos contextos, hasta los procesos de admisión en instituciones de enseñanza. En este nivel se exige cumplir una serie de parámetros que garanticen el trato justo al ciudadano y que serían validados en auditorías específicas. Informar a los usuarios, documentar adecuadamente los algoritmos y contar con data sets de calidad son algunos de estos requerimientos.

Para los sistemas que no entran en estas categorías, denominados de bajo riesgo, la legislación también impone algunas obligaciones, principalmente vinculadas a la transparencia de los algoritmos, siempre que se trate de sistemas que interactúan con humanos, realizan algún tipo de identificación biométrica o de sentimientos, o generan contenidos falsos.

Al margen de esta legislación específica, conviene tener en cuenta otros dos aspectos fundamentales: el cumplimiento de las leyes de protección de datos (LOPD) y, en el caso de los sistemas de generación de contenidos, de las leyes de copyright.

Respecto a las leyes de protección de datos, los puntos más importantes a considerar es que ciertos datos no pueden procesarse sin un consentimiento expreso del cliente al que perteneces. Estas categorías protegidas, denominados datos sensibles[27], incluyen datos sanitarios, confesionales, de origen o de vida sexual.

5.3. Casos de uso frecuentes

La gran variedad existente en los algoritmos de Machine Learning permite que éstos se puedan aplicar en muchos casos de uso diferentes. En esta sección trataremos de resumir algunos de los problemas más habituales que se abordan con estas técnicas.

Predicción y clasificación

El uso más evidente de los algoritmos de machine learning es el de emular el criterio humano en la toma de decisiones que no se ajustan a unas reglas de negocio, o bien predecir el

[27] Una explicación sobre datos sensibles por parte de la empresa de protección de datos Atico34 en este enlace: https://protecciondatos-lopd.com/empresas/datos-especialmente-protegidos-sensibles/

comportamiento de una variable en función de sus valores pasados. Dentro de este amplio grupo de funcionalidades, podemos ubicar los siguientes grandes casos de uso:

- **Clasificación de clientes**: los algoritmos de clasificación permiten identificar grupos de clientes dentro de nuestra compañía, bien sea para asociar propensiones de compras de algunos productos, para dirigir campañas, para identificar riesgos de fuga o para establecer segmentos de clientes.
- **Predicción de volúmenes de ventas u operaciones**: los algoritmos en series temporales pueden utilizarse para predecir volúmenes futuros de operaciones, facilitando la gestión de recursos o de stock de la compañía.
- **Análisis de riesgos**: especialmente relevante en sectores como el financiero o el asegurador, los algoritmos de ML pueden emular el comportamiento humano analizando riesgos y levantando alertas o automatizando acciones en ciertos escenarios.
- **Detección de fraude**: los algoritmos de identificación de anomalías son especialmente relevantes a la hora de identificar potenciales operaciones fraudulentas.
- **Pricing**: los algoritmos de ML pueden proporcionar información relevante que afecte al pricing de ciertos productos de forma dinámica. Combinando distintos algoritmos, podemos crear un sistema de pricing que considere la elasticidad al precio del cliente, su riesgo de fuga o impago, la situación del mercado y otros factores que ayuden a optimizar el precio de nuestros productos.

Hiperautomatización

La *hiperautomatización* es una de las macrotendencias del momento, de acuerdo con lo expuesto por Gartner[28]. Lo que se esconde detrás del término es la automatización de procesos de negocio aprovechando capacidades de inteligencia artificial. Una de las tecnologías básicas para lograr este objetivo es la automatización robótica de procesos (RPA, por sus siglas en inglés), que no es algo nuevo, pero sí se ha sofisticado mucho en los últimos años.

RPA consiste en la creación de sistemas software capaces de interactuar con un ecosistema de aplicaciones del mismo modo que lo haría una persona, es decir, utilizando las propias interfaces de los sistemas implicados. La ventaja frente a la automatización de sistemas habitual es que es muy poco invasiva, no requiere modificar las aplicaciones con las que interactúa, sino que se relaciona directamente con ellas, exactamente como haría una persona (de hecho, al ver un robot RPA en acción, la percepción es que el ordenador en el que actúa se está ejecutando de forma remota, solo que mucho más rápido de lo normal).

RPA es una forma rápida y barata de automatizar tareas, pero presenta limitaciones, fundamentalmente que tienen una gran dependencia de los sistemas con los que interactúan (un cambio en una interfaz puede hacer que los robots que utilizan esa herramienta dejen de funcionar) y que sólo puede automatizar procesos reglados, donde todas las decisiones se puedan expresar en reglas inequívocas, y utilizando datos de entrada estructurados. En este segundo punto es donde entra en juego la *hiperautomatización*: utilizando tecnologías de Inteligencia

[28] Más sobre la visión de Gartner de la *hiperautomatización* aquí: https://www.gartner.com/doc/reprints?id=1-27U4ZXLL&ct=211101&st=sb

Artificial se pueden tomar decisiones más inteligentes, basadas en modelos de ML, y se pueden procesar datos complejos (textos en lenguaje natural, documentos escaneados o fotografías).

La combinación del procesamiento de datos no estructurados, la toma de decisiones basada en modelos predictivos y la actuación sobre los sistemas core de la compañía a través de tecnologías como RPA, da lugar a un potencial casi ilimitado de automatización de procesos que prácticamente puede aprovecharse en cualquier sector.

Asistentes conversacionales

Otro de los ámbitos de aplicación más habitual de la IA es la atención al cliente. En realidad, es un ámbito completamente relacionado con la *hiperautomatización*, ya que el principal objetivo que buscan las compañías con estas soluciones es optimizar los costes de sus servicios de atención al cliente, pero la realidad es que la IA puede ir más allá, consiguiendo mejorar la experiencia del cliente y ofreciendo capacidades que, en el mundo actual, muchos clientes ya esperan de cualquier servicio.

La necesidad de automatización de la atención al cliente también está íntimamente ligada con el concepto de omnicanalidad. Los clientes actuales quieren poder contactar con empresas y acceder a sus servicios a través de cualquier canal (teléfono, aplicaciones, web, whatsapp, correo electrónico, presencialmente en una oficina). Las empresas nativas digitales a menudo limitan sus servicios a canales digitales, lo que puede simplificar un poco el problema, pero también obliga a ofrecer una experiencia de cliente completa en dichos canales. Por su parte, las empresas con una herencia tradicional se ven obligadas a compatibilizar la existencia de todos estos canales de forma simultánea. La inteligencia artificial entra al servicio de esta necesidad.

Empezando por el canal telefónico, prácticamente la totalidad de las empresas cuentan ya con sistemas informáticos capaces de interpretar la necesidad del cliente que llama, dejando que éste lo exprese con sus propias palabras (es decir, evitando los menús de marcación por tonos). Para hacer esto, se combinan hasta tres tecnologías diferentes: *voice-to-text* para transcribir la petición del cliente a un texto, NLP para determinar la intención de ese texto y *text-to-speech* para convertir la respuesta del sistema en voz (realmente, este último paso se puede sustituir por locuciones, depende de lo fijos que sean los mensajes con los que se va a contestar). Estos sistemas se utilizan para automatizar ciertas operativas y, en otros casos, recabar toda la información posible antes de transferir la llamada a un agente.

Pero el uso de la IA no termina ahí. También son habituales las herramientas de agentes de *contact center* que cuentan con un soporte basado en IA que les ayuda a atender la llamada, bien sea ofreciendo guiones con la “next best action” recomendada por un algoritmo de ML, o aplicando detección de emociones a la conversación para poder alertar de un posible riesgo de fuga del cliente. Estas llamadas también pueden analizarse a posteriori, enriqueciendo la visión que se tiene del cliente.

Como veíamos al hablar de los sistemas conversacionales, el mismo reconocimiento de intenciones aplicado a las llamadas telefónicas se puede llevar a canales escritos, adaptando la

conversación al canal. De esta manera, se puede crear una experiencia de cliente única a través de los distintos canales de la compañía y, en todos ellos, se podrá mantener la misma visión del cliente, ya que cada interacción alimentará nuestro CRM con toda la información que resulte relevante.

Por último, la IA permite facilitar el reconocimiento del cliente al entrar en nuestros sistemas, a través de parámetros biométricos, como la huella dactilar en dispositivos móviles, o incluso el reconocimiento de la voz en algunos casos. Esto sumado a la facilidad de contacto propiciada por la omnicanalidad, siempre que esta se diseñe centrada en el cliente, repercute en una mejor experiencia de cliente, reduciendo riesgo de fuga y aumentando las opciones de up o *cross selling*.

5.4. Herramientas de ML

Las herramientas de ML que se utilizan en la actualidad facilitan mucho la creación de modelos, poniéndolos al alcance prácticamente de cualquiera, si bien un científico de datos experto siempre será capaz de sacar un rendimiento mucho mayor a estas tecnologías.

Para empezar, los lenguajes de programación que más se utilizan en el mundo del ML, como **Python** o **R**, cuentan con bibliotecas preparadas con los principales algoritmos ya programados, de manera que el científico de datos que las emplee no tiene que reinventar la rueda y codificar las fórmulas matemáticas que se encuentran detrás de esos algoritmos, sino "únicamente" preocuparse por entrenarlos adecuadamente con juegos de datos. Pero ni siquiera es necesario aprender a programar para trabajar con algoritmos de ML, ya que existe actualmente un amplio abanico de herramientas especializadas con distinto grado de facilidad de uso, desde herramientas pensadas para ahorrar tiempo a un experto en IA, hasta herramientas que pueden ser utilizadas por usuarios de negocio. En el gráfico adjunto, podemos ver las que, para **Gartner**, eran las principales plataformas de ML del mercado en enero de 2021.

Una de las principales tendencias tecnológicas de la actualidad consiste en acercar cada vez más el uso de herramientas software de relativa complejidad a los usuarios de negocio. Ocurre, por ejemplo, con las herramientas ***self-service* de *Business Intelligence*** (BI): tradicionalmente, para generar un nuevo informe de BI, un usuario de negocio tenía que solicitar una actuación por parte del equipo de informática, sin embargo, nuevas herramientas como **Power BI**, **Tableau** o **QlikView**, permiten la configuración

sencilla de nuevos informes con datos en tiempo real a través de una interfaz amigable.

Esta tendencia se hace más evidente en las llamadas herramientas "**no-code**" o "**low-code**[29]", que permiten que usuarios sin experiencia en programación, puedan crear o parametrizar aplicaciones sin usar líneas de código a niveles que hace unos años requerirían de la intervención del equipo de IT. **BigML** o **Obviouly.ai** son ejemplos de este tipo de herramientas, con ella se pueden crear modelos de ML de manera sencilla, a través de una interfaz visual que permite incorporar datos, visualizarlos, crear y parametrizar modelos y evaluar los resultados, todo ello sin escribir una sola línea de código. El "no-code" está llegando mucho más lejos, de hecho, apoyándose en la IA, ya que empiezan a existir sistemas capaces de crear un código a partir de una descripción en lenguaje natural o crear un formulario web a partir de un boceto dibujado en papel[30].

¿Significa esto que no es necesario saber nada de IA para crear modelos de IA? No, es importante entender lo que se está haciendo para evitar entrenar un modelo deficiente. Además, el rendimiento de los modelos realizados de forma automatizada sigue siendo muy inferior al que puede obtener un científico de datos con un preprocesamiento adecuado de los datos y la elección correcta del algoritmo y sus parámetros.

5.5. Principales players del mercado

Al hablar de principales empresas que trabajan en el mundo de ML, es inevitable pensar en las grandes compañías tecnológicas como Google, Microsoft o Amazon. Si embargo, existen infinidad de compañías de nicho especializas en algunos servicios en concreto, siendo prácticamente imposible estar siempre actualizado.

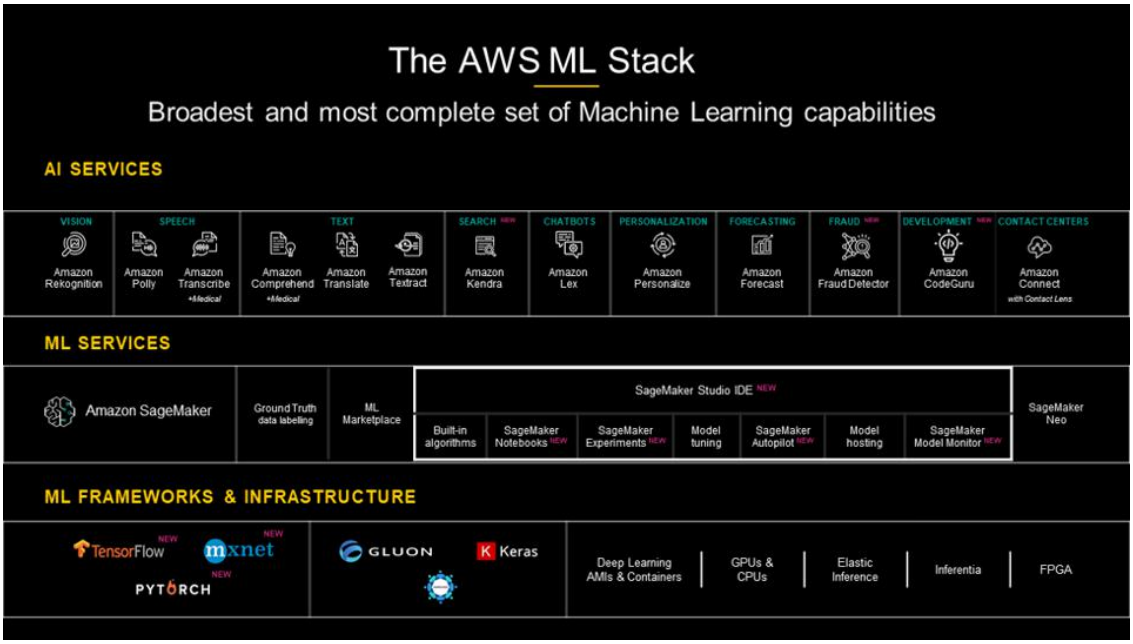
Grandes tecnológicas

Desde su irrupción en el mundo *cloud*, Google, Microsoft y Amazon, a través de sus plataformas Google Cloud, Azure y AWS, han mantenido una carrera por convertirse en los principales proveedores de servicios tecnológicos en la nube, ofertando desde capacidades de infraestructura (mencionábamos brevemente el IaaS en el capítulo de introducción de estos apuntes), como servicios más específicos. Hemos mencionado también cómo Google se definía ya en 2016 como una empresa *Machine Learning First* y esto se ve reflejado en su oferta de servicios.

Dentro de las plataformas mencionadas, se puede encontrar servicios que facilitan el desarrollo y despliegue de capacidades de ML (TensorFlow en Google, Azure ML Studio en Microfost y Amazon SageMaker en AWS), pero también servicios específicos para otras funciones, como el desarrollo de asistentes conversacionales, capacidades de OCR, *Voice to text* y *Text to speech*, etc.

[29] En este enlace podéis encontrar la visión de Microsoft de las herramientas low-code y no-code: https://powerapps.microsoft.com/en-us/low-code-no-code-development-platforms/
[30] Power Platform está desarrollando esta característica, más información aquí: https://learn.microsoft.com/es-es/power-apps/maker/canvas-apps/app-from-image

Desenvolverse bien con estas plataformas es una puerta de entrada segura para el desarrollo de la Inteligencia Artificial a nivel empresarial. Bajo estas líneas podéis ver el catálogo de servicios de AI y ML que ofrece AWS, en el que se incluyen también *frameworks* de terceros, como TensorFlow.



También cabe destacar el papel que Open IA está jugando en capacidades disruptivas de generación de contenidos, con servicios que ya hemos mencionado, como ChatGPT y DALL-E. Recientemente, en enero de 2023, Microsoft reforzó su *partnership* con esta compañía para integrar sus servicios dentro de la plataforma de Azure[31], que también es la plataforma que Open AI utiliza en su desarrollo. Además, ya estamos empezando a ver cómo estas capacidades se integran también en su suite de ofimática a través de Copilot[32].

Compañías de nicho

Los grandes players que hemos mencionado son, posiblemente, la mejor manera de desarrollar ML internamente dentro de una empresa, siempre y cuando se cuente con el conocimiento para poder trabajar con estas herramientas. No obstante, existen numerosas compañías de nicho, en algunos casos start-ups, especializadas en soluciones concretas, que pueden ser una buena opción para abordar algunos de los casos de uso que hemos mencionado, sin necesidad de desarrollarlo desde cero, o bien como apoyo para el desarrollo de soluciones internas. En los ecosistemas de desarrollo de soluciones de IA, es habitual trabajar con un pipeline de productos que faciliten el desarrollo en sus distintas etapas, desde la consecución y preprocesamiento de los datos, hasta la verificación de los modelos, y las plataformas ofrecidas por los grandes fabricantes del mercado cuentan con capacidades para integrar estas soluciones de terceros en sus arquitecturas.

[31] Microsoft explica este acuerdo en este enlace: https://blogs.microsoft.com/blog/2023/01/23/microsoftandopenaiextendpartnership/
[32] Una explicación sobre qué es Copilot dentro de la web de Microsoft: https://news.microsoft.com/reinventing-productivity/

A continuación, mencionaremos algunos ejemplos[33] de este tipo de compañías, únicamente para dar una idea del tipo de soluciones que pueden ofrecer.

- **Redflag AI**: compañía especializada en identificar potenciales violaciones de copyright, a través de procesamiento de texto, imágenes y sonido en la web. Su solución es capaz de analizar datos públicos, incluidas redes sociales.
- **Tonic.ai**: una compañía especializada en la generación de datos falsos (datos sintéticos) útiles para el entrenamiento de otros algoritmos. Su solución emplea redes neuronales para producir datos realistas que simulen diversas condiciones para alimentar otros modelos.
- **Flyr**: ofrece una herramienta capaz de afinar los modelos de pricing de compañías aéreas, incorporando ML en la toma de decisiones.
- **Interactions**: centrada en el desarrollo de asistentes virtuales, permite a las compañías usuarias atender a sus clientes a través de canales de voz y de texto, interpretando sus interacciones a través de ML.
- **Databricks**: es una de las plataformas más relevantes para el desarrollo de soluciones de inteligencia artificial en el mundo empresarial, facilitando la creación de espacios colaborativos de desarrollo y la gestión del ciclo de vida de las soluciones de machine learning.

Actualmente, la inteligencia artificial y, más en particular, machine learning, es una de las principales tendencias del mercado en todos los sentidos: tanto por su poder revulsivo dentro del desempeño de las compañías, como por su propio potencial como producto en sí mismo. El mercado de soluciones de ML continuará creciendo en los próximos años y ser consciente de su evolución y novedades puede suponer una ventaja competitiva crucial en el desarrollo de cualquier empresa.

[33] Un listado de 40 compañías de machine learning interesantes: https://builtin.com/artificial-intelligence/machine-learning-companies

Bibliografía y recursos

Susanne Chishti, Ivana Bartoletti, Anne Leslie, Shân M. Millie (2020). *The AI Book: The Artificial Intelligence Handbook for Investors, Entrepreneurs and FinTech Visionaries*. Fintech Circle.

Tom Taulli (2019). *Artificial Intelligence Basics: A Non-Technical Introduction*. Apress.

Nick Bostrom (2014). *Superintelligence: Paths, Dangers, Strategies*. Oxford University Press.

Steven Finlay (2018), *Artificial Intelligence and Machine Learning for Business*. Relativistic.

Andriy Burkov (2019), The hundred-page Machine Learning book. Andriy Burkov.