

Técnicas avanzadas y evaluación del modelo.



Caso práctico



[DCStudio \(CC BY-SA\)](#)

Max es una estudiante de inteligencia artificial que ya ha terminado las prácticas que estaba haciendo en una empresa. Tiene algunas dudas sobre un par de casos que ha visto en sus prácticas y que no le quedaron del todo claras hablando con su

responsable en la empresa.

Ha quedado hoy con uno de sus profesores, Antonio, y éste ha propuesto que hagan una llamada grupal para revisarlo entre todos.

"¿Cómo sé que el modelo que he entrenado es bueno?" Pregunta Max "A veces hago pruebas con distintos parámetros y salen modelos con resultados diferentes. ¿Con cuál debería quedarme?"

Antonio le contesta "Pues como se hace con cualquier otro producto... Tienes que medir y comparar"

Todos sonríen y se remueven en la silla hasta que Max se atreve a preguntar "Ok, pero ¿qué medimos?"

"Tranquila, hay una serie de métricas que nos van a ser útiles para ésto. Os enseñaré algunos ejemplos" Dice Antonio compartiendo pantalla para enseñarles un desarrollo de ejemplo.

Llegamos a la última unidad del módulo en la que nos centraremos en evaluar si el modelo es realmente preciso y podemos fiarnos de las predicciones que haga. Esto lo vamos a poder hacer gracias las técnicas:

- ✓ Métricas de evaluación en aprendizaje automático.
- ✓ Búsqueda de la mejor combinación de hiperparámetros de los modelos.
- ✓ Tratamiento de conjuntos de datos no balanceados.
- ✓ Detección de anomalías.

Haremos una revisión metodológica de estas técnicas, que luego se llevan a la práctica de diferentes formas según el ecosistema en el que se trabaje.



[Ministerio de Educación y Formación Profesional](#) (Dominio público)

Materiales formativos de FP Online propiedad del Ministerio de Educación y Formación Profesional.

[Aviso Legal](#)

1.- Evaluación del modelo.



Caso práctico



[DCStudio \(CC BY-SA\)](#)

Max ha tomado nota de las métricas y métodos que ha nombrado Antonio para evaluar si el modelo es el más preciso o realmente se puede fiar de las predicciones que arroje.

Decide probarlo con un problema que ha estado analizando y modelizando con diversas herramientas. Se descarga el archivo csv de las predicciones

que le daba la herramienta que ha estado utilizando para un conjunto de datos sobre tasa de abandono de clientes y se dispone a compararlas con los datos reales que tenía.

¿Qué exactitud y sensibilidad habrá alcanzado su modelo ajustado?



[storyset \(CC BY-SA\)](#)

Existen varias métricas que nos permiten evaluar cómo de bueno es el modelo de aprendizaje automático que tenemos, tras el ajuste o entrenamiento. Como los modelos más comunes son los de aprendizaje automático supervisado, nos vamos a centrar en éstos.

Dentro del aprendizaje automático supervisado, es decir, los modelos en los que tenemos los casos "etiquetados" o con su variable de salida correcta correspondiente, hay dos tipos de problemas: clasificación y regresión.

Evaluación en problemas de clasificación

Imagina el caso de una compañía telefónica que quiere predecir si un cliente se va a dar de baja en los próximos 30 días. Una vez entrenado el modelo, lo probamos con un conjunto de datos de test que habíamos reservado y que no han sido utilizados en el entrenamiento. La tabla resultante de dicho test podría ser ésta:

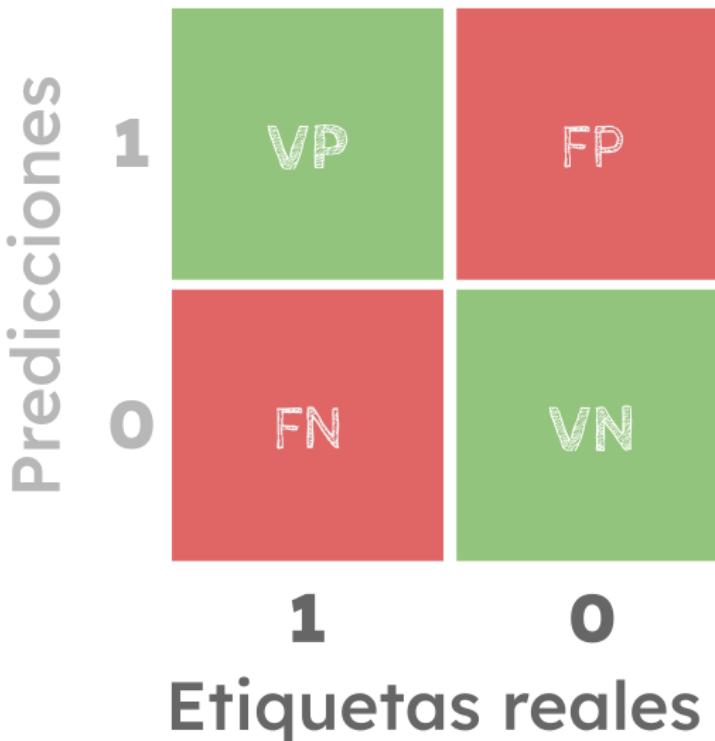
ID	Predicción	Etiqueta real	Error	Resultado
1	0	0	0	VN
2	0	0	0	VN
3	0	0	0	VN
4	1	1	0	VP
5	0	1	1	FN
6	0	0	0	VN
7	0	0	0	VN
8	0	0	0	VN
9	1	0	1	FP
10	0	0	0	VN
11	0	0	0	VN
12	0	0	0	VN
13	0	0	0	VN
14	1	0	1	FP
15	0	0	0	VN
16	0	0	0	VN
17	0	0	0	VN
18	0	0	0	VN
19	1	1	0	VP
20	0	0	0	VN

Fran Bartolomé - Elaboración propia (Dominio público)

En ella vemos que hay valores de 1 y 0 que corresponden a si el cliente se da de baja o no respectivamente. En los casos en que no coincide la predicción con la etiqueta real que se tenía, se contabiliza el error según esta nomenclatura:

- ✓ VN: verdadero negativo.
- ✓ VP: verdadero positivo.
- ✓ FN: falso negativo.
- ✓ FP: falso positivo.

El primer recurso que podemos utilizar, muy visual, es lo que se denomina matriz de confusión. Tiene este aspecto:



Fran Bartolomé - Elaboración propia (Dominio público)

Es una herramienta ampliamente utilizada que permite inspeccionar y evaluar visualmente las predicciones de nuestro modelo. En cada fila se representa el número de predicciones de cada clase y en las columnas las instancias de la clase real.

Para el ejemplo que hemos planteado, se representaría así:

1	2	2
0	1	15
	1	0

Las métricas cuyos valores nos indican la calidad del modelo son:

- ✓ **Exactitud o accuracy:** la fracción de predicciones que el modelo realizó correctamente. Se representa como un porcentaje o un valor entre 0 y 1. Es una buena métrica cuando tenemos un conjunto de datos balanceado, esto es, cuando el número de etiquetas de cada clase es similar. La exactitud de nuestro modelo de ejemplo es de 0,85, ya que ha acertado 17 predicciones de 20. El problema es que si, en vez de haber dado un falso negativo y dos falsos positivos, hubiese dado tres falsos negativos, la exactitud sería la misma, pero estaríamos perdiendo más clientes sin saberlo. Por eso necesitamos métricas que nos den más información sobre los distintos errores.
- ✓ **Recall o sensibilidad:** indica la proporción de ejemplos positivos que están identificados correctamente por el modelo entre todos los positivos reales. Es decir, $VP / (VP + FN)$. En nuestro ejemplo, el valor de sensibilidad sería $2 / (2 + 1) = 0,67$. Si evaluásemos con esta métrica un modelo que siempre prediga la etiqueta positiva ("Sí") tendría una sensibilidad casi de 1, pero no sería un modelo demasiado inteligente. Aunque lo ideal para nuestro modelo es maximizar la sensibilidad, esta métrica por sí sola no nos asegura que tengamos un buen modelo.

- ✓ **Precisión:** esta métrica está determinada por la fracción de elementos clasificados correctamente como positivo entre todos los que el modelo ha clasificado como positivos. La fórmula es $VP / (VP + FP)$. El modelo de ejemplo tendría una precisión de $2 / (2 + 2) = 0.5$. Volvamos ahora al modelo que siempre predice la etiqueta positiva. En ese caso, la precisión del modelo es $2 / (2 + 17) = 0.1$. Vemos como este modelo tenía una sensibilidad máxima, pero tiene una precisión muy pobre. En este caso necesitamos de las dos métricas para evaluar la calidad real del modelo.
- ✓ **F1 score:** combina las métricas Precision y Recall para dar un único resultado. Esta métrica es la más apropiada cuando tenemos conjuntos de datos no balanceados. Se calcula como la media armónica de Precision y Recall. La fórmula es $F1 = (2 * precision * recall) / (precision + recall)$. Quizá te preguntes por qué la media armónica y no la simple. Esto es porque la media armónica hace que si una de las dos medidas es pequeña (aunque la otra sea máxima), el valor de F1 score va a ser pequeño. En nuestro caso, obtenemos un valor de $F1 = 0.57$.

Evaluación en problemas de regresión.

A diferencia de los modelos de clasificación, en los modelos de regresión es casi imposible predecir el valor exacto, sino que más bien se busca estar lo más cerca posible del valor real, por lo que la mayoría de las métricas, con sutiles diferencias entre ellas, van a centrarse en medir eso: lo cerca (o lejos) que están las predicciones de los valores reales.

En este caso, tenemos como ejemplo las predicciones de un modelo que determina el precio de relojes dependiendo de sus características. En la tabla mostramos el precio predicho por el modelo, el precio real, el error absoluto y el error elevado al cuadrado.

ID	Predicción	Valor real	Error absoluto	Error cuadrático
1	12	10	2	4
2	300	800	500	250000
3	62	80	18	324
4	55	51	4	16
5	50	35	15	225

Fran Bartolomé - Elaboración propia (Dominio público)

Algunas de las métricas de evaluación más comunes para los modelos de regresión son:

- ✓ **Error medio absoluto:** Es la media de las diferencias absolutas entre el valor objetivo y el predicho. Al no elevar al cuadrado, no penaliza los errores grandes, lo que la hace no muy sensible a valores anómalos, por lo que no es una métrica recomendable en modelos en los que se deba prestar atención a éstos. Esta métrica también representa el error en la misma escala que los valores reales. Lo más deseable es que su valor sea cercano a cero. Para nuestro modelo de cálculo de precios de relojes, el error medio absoluto es 107.8.
- ✓ **Media de los errores al cuadrado (error cuadrático medio):** Una de las medidas más utilizadas en tareas de regresión. Es simplemente la media de las diferencias entre el valor objetivo y el predicho al cuadrado. Al elevar al cuadrado los errores, magnifica los errores grandes, por lo que hay que utilizarla con cuidado cuando tenemos valores anómalos en nuestro conjunto de datos. Puede tomar valores entre 0 e infinito. Cuanto más cerca de cero esté la métrica, mejor. El error cuadrático medio del modelo de ejemplo es 50113.8. Vemos como en el caso de nuestro ejemplo se magnifican los errores grandes.

- ✓ **Raíz cuadrada de la media del error al cuadrado:** Es igual a la raíz cuadrada de la métrica anterior. La ventaja de esta métrica es que presenta el error en las mismas unidades que la variable objetivo, lo que la hace más fácil de entender. Para nuestro modelo este error es igual a 223.86.
- ✓ **R cuadrado:** también llamado coeficiente de determinación. Esta métrica difiere de las anteriores, ya que compara nuestro modelo con un modelo básico que siempre devuelve como predicción la media de los valores objetivo de entrenamiento. La comparación entre estos dos modelos se realiza en base a la media de los errores al cuadrado de cada modelo. Los valores que puede tomar esta métrica van desde menos infinito a 1. Cuanto más cercano a 1 sea el valor de esta métrica, mejor será nuestro modelo. El valor de R cuadrado para el modelo será de 0.455.
- ✓ **R cuadrado ajustado:** es una mejora de R cuadrado. El problema de la métrica anterior es que cada vez que se añaden más variables independientes (o variables predictoras) al modelo, R cuadrado se queda igual o mejora, pero nunca empeora, lo que puede llegar a confundirnos, ya que, porque un modelo utilice más variables predictoras que otro, no quiere decir que sea mejor. R cuadrado ajustado compensa la adición de variables independientes. El valor de R cuadrado ajustado siempre va a ser menor o igual al de R cuadrado, pero esta métrica mostrará mejoría cuando el modelo sea realmente mejor. Para esta medida no podemos hacer el cálculo para nuestro modelo de ejemplo porque, como hemos visto antes, depende del número de ejemplos y el número de variables utilizadas para entrenar dicho modelo.

A la hora de trabajar con algoritmos de aprendizaje supervisado es muy importante la elección de una métrica de evaluación correcta para nuestro modelo. Para los modelos de clasificación es muy importante prestar atención al conjunto de datos y comprobar si es balanceado o no. En los modelos de regresión hay que considerar los valores anómalos y si queremos penalizar errores grandes o no.

No obstante, generalmente, el dominio de negocio será el que nos guíe en la correcta elección de la métrica. Por ejemplo, hay casos de clasificación en los que no nos podremos permitir dejar pasar valores positivos como falsos negativos. Y en algunos casos de regresión, puede que no nos podamos permitir alejarnos mucho de los valores reales, por lo que elegir la media de los errores al cuadrado nos ayudaría a penalizar los errores grandes.



Autoevaluación

El Error Medio [REDACTED] es la media de las diferencias absolutas entre el valor objetivo y el predicho

Se trata del error medio absoluto

1.1.- Validación cruzada.



[vectorjuice \(CC BY-SA\)](#)

Cuando evaluamos el modelo, estamos probando el modelo en un conjunto concreto de datos, que además suele ser pequeño. Para obtener un valor de exactitud o precisión lo más general posible, en realidad, debemos aplicar alguna de las técnicas que diversifican el entrenamiento y la evaluación de formas más elaboradas sobre el dataset que tenemos. De esta forma, nos aseguramos de que los resultados son independientes de la partición entre datos de entrenamiento y prueba que hemos realizado en un momento puntual. La evaluación puede depender en gran medida de cómo es esa división y, debido a estas carencias aparece el concepto de validación cruzada

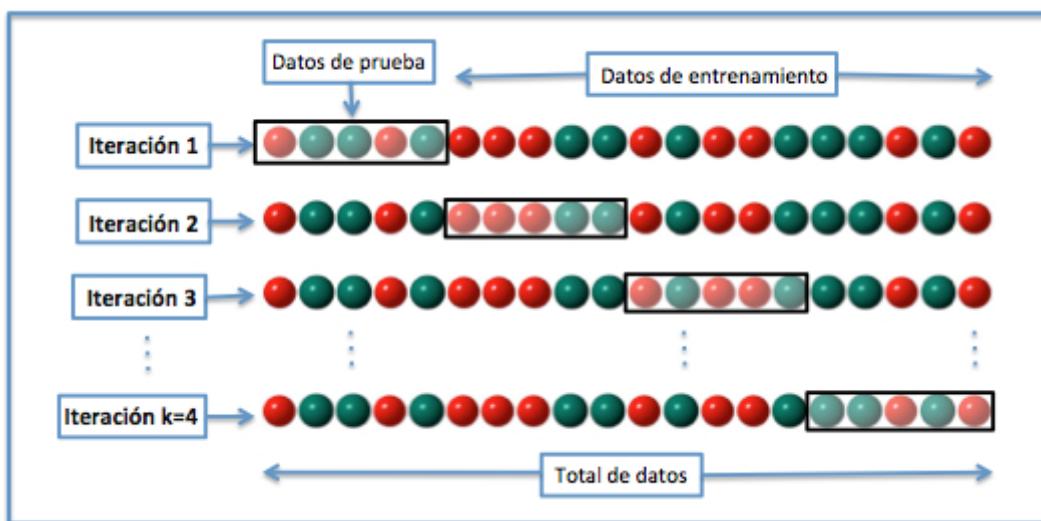
Consiste en repetir y calcular la media aritmética obtenida de las medidas de evaluación sobre diferentes particiones. Las diferentes formas de aplicar este proceso y de hacer las particiones, da lugar a las diferentes técnicas de validación cruzada.

Validación cruzada de K iteraciones

En la validación cruzada de K iteraciones o K-fold cross-validation los datos con los que contamos se dividen en K subconjuntos. Entonces se inician una serie de iteraciones en cada una de las cuales se utilizará uno de los subconjuntos como datos de test y el resto como datos de entrenamiento del modelo. Al final, se calcula la media aritmética de todos los resultados de la métrica.

La elección de K o del número de iteraciones depende de la medida del conjunto de datos. Lo más común es utilizar la validación cruzada de 10 iteraciones (10-fold cross-validation).

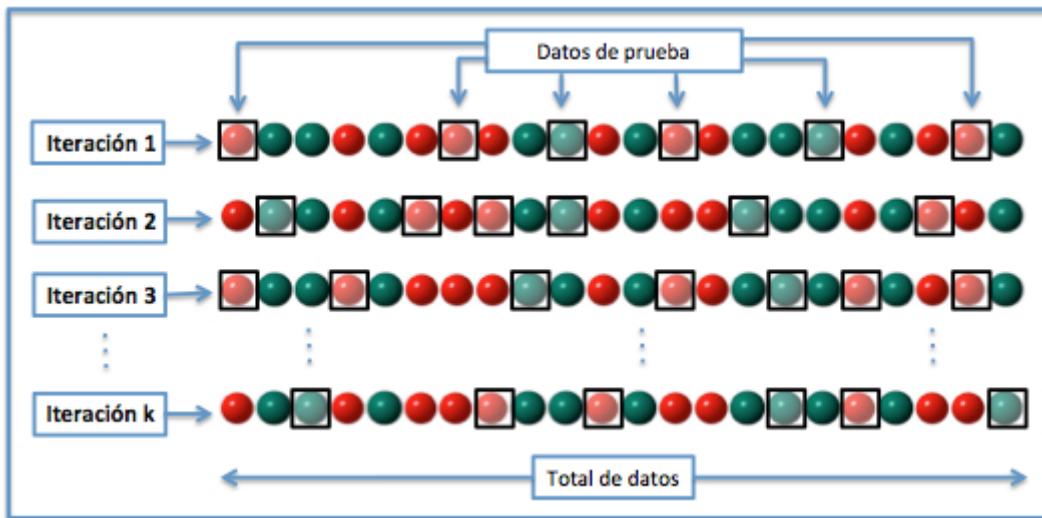
El inconveniente de esta técnica es que hace que el proceso de evaluación sea más lento, porque multiplica el tiempo de cálculo de la métrica por K veces.



[Joan.domenech91 \(CC BY-SA\)](#)

Validación cruzada aleatoria

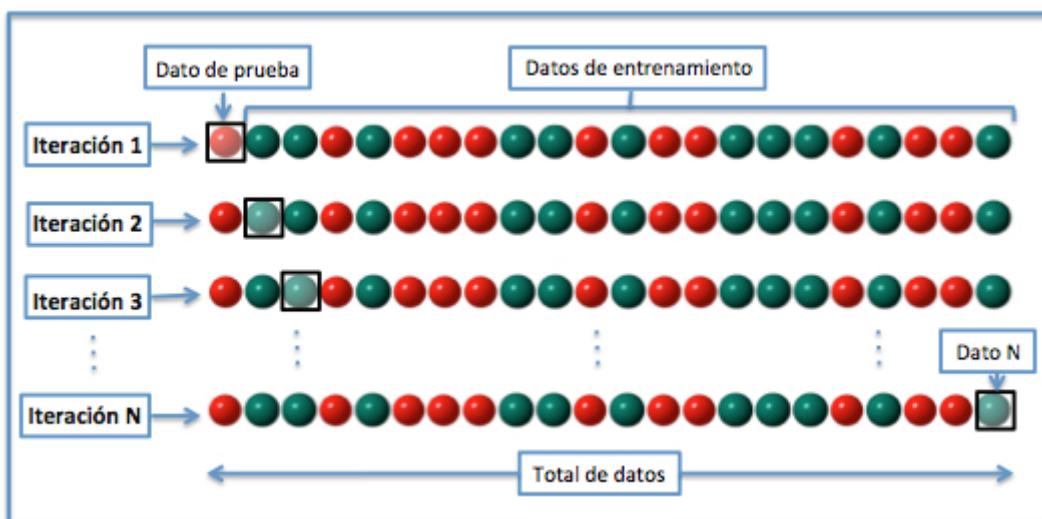
En este segundo enfoque, el bloque de validación se escoge aleatoriamente, repitiéndose el proceso k veces (siendo tanto el tamaño de cada bloque como el número k cifras arbitrarias). La ventaja de este método es que el número de iteraciones (k) no depende del tamaño de los subbloques que se consideren. La desventaja es que no es posible asegurar que todas las muestras van a ser consideradas como parte de los conjuntos de entrenamiento o de validación (pues habrá muchas que no sean escogidas nunca), y que habrá muestras que puedan ser consideradas en más de una iteración.



[Joan.domenech91 \(CC BY-SA\)](#)

Validación cruzada dejando uno fuera

En este tercer tipo de validación cruzada, cada registro u observación será la muestra para validar o para test, dejando todo el resto de muestras ($N-1$) para el entrenamiento. La ventaja es que tenemos un gran conjunto de datos disponible para el entrenamiento, pero la desventaja obvia es que va a ser un proceso muy, muy costoso computacionalmente. No obstante, hay casos en los que es necesario por el riesgo de overfitting o sobre ajuste.



[Joan.domenech91 \(CC BY-SA\)](#)

2.- Optimización de hiperparámetros.



Caso práctico



[DCStudio \(CC BY-SA\)](#)

Max está un poco bloqueada. Ya ha evaluado su modelo y lo cierto es que no sabe muy bien si dejarlo como está o hacer pruebas y pruebas hasta conseguir mejores métricas. Se mete en el foro de dudas de la plataforma que usan en su curso, con la intención de volver a hacer preguntas a su profesor, pero se encuentra con una duda muy parecida a la suya.

En el hilo de respuestas encuentra de forma recurrente la expresión "optimización de hiperparámetros" como una técnica que permite, en algunos casos, encontrar buenas configuraciones en los modelos más conocidos. No siempre es sencillo de aplicar, y a veces es contraproducente, pero a Max le parece muy interesante que haya un recurso más que probar y decide investigar más sobre ello.

Cada algoritmo de aprendizaje automático suele tener varios parámetros que es necesario elegir y fijar para que la arquitectura de éste quede definida. No nos referimos a las parámetros internos del modelo, los que se ajustan de forma automática durante el entrenamiento, sino a los que dotan de una cierta configuración al modelo. Por ejemplo, en el caso de un modelo KNN, al instanciar la clase, debemos elegir el valor de K, o en el modelo de árbol de decisión, el parámetro de la profundidad máxima de sus bifurcaciones también afecta a la precisión de éste.

A estos parámetros especiales se les conoce como los hiperparámetros del modelo y su valor se utiliza para controlar el proceso de aprendizaje. La optimización o ajuste de hiperparámetros es el problema de elegir el conjunto de éstos que maximiza la tasa de acierto del modelo.

La optimización de hiper-parámetros se realiza normalmente mediante la utilización de un proceso de búsqueda cuyo objetivo consiste en encontrar la mejor selección de valores para un conjunto finito de hiper-parámetros con el objetivo de generar el mejor modelo posible.

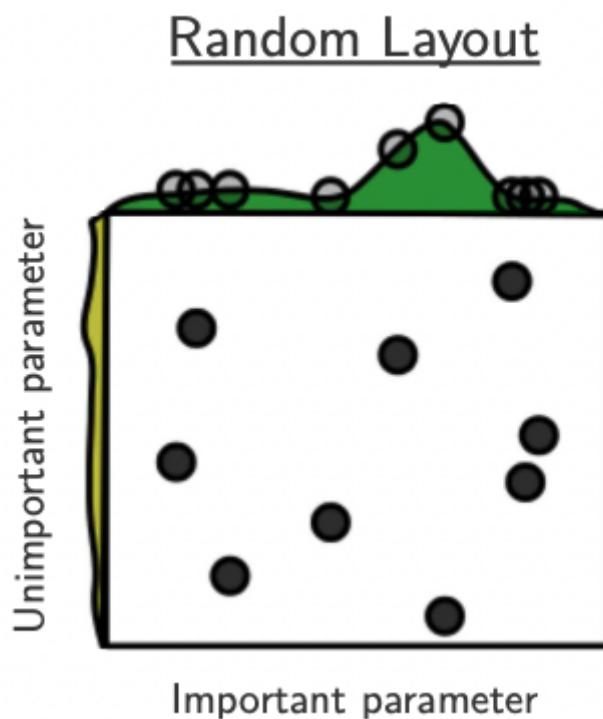
Desde la perspectiva de un proceso de optimización de hiper-parámetros, el espacio de estados se corresponde con todas las posibles configuraciones de los hiper-parámetros en base a los rangos de posibles valores definidos para cada uno de ellos; las acciones se corresponden con operaciones utilizadas por el algoritmo de búsqueda para generar nuevas estados; y la función objetivo que permite guiar el proceso de búsqueda con el objetivo de maximizar o minimizar los valores obtenidos para cada estado. Es muy común utilizar el

“accuracy” obtenido tras la ejecución de la fase de entrenamiento como valor objetivo para guiar el proceso de búsqueda.

Como se puede deducir, este proceso es normalmente muy costoso desde el punto de vista computacional, ya que supone la ejecución de múltiples fases de entrenamiento mediante la modificación conjunta de los diferentes hiper-parámetros a optimizar. Es decir, cuanto mayor sea el número de hiper-parámetros a seleccionar, mayor será el tiempo y el coste computacional del proceso de optimización debido al elevado número de combinaciones válidas que pueden seleccionarse.

Búsqueda aleatoria (Random Search)

Es un proceso de búsqueda de tipo aleatorio sobre un espacio de búsqueda finito. En el caso de los hiper-parámetros el proceso de búsqueda aleatoria consiste en generar nuevos estados (configuraciones de hiper-parámetros) mediante la modificación aleatoria de una de las soluciones previamente generadas dentro del espacio de búsqueda, siendo normalmente la que mejor valor ha obtenido tras aplicar la función objetivo aunque, dependiendo de la implementación del algoritmo, se genera la nueva solución a partir del último estado (configuración de hiper-parámetros).



[Bergstra&Bengio \(CC BY-SA\)](#)

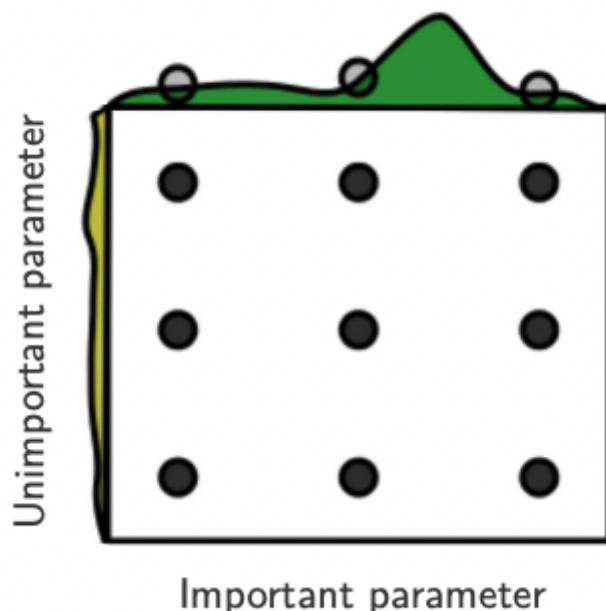
En la imagen podemos observar la representación espacial de un proceso de búsqueda aleatoria para diferentes hiper-parámetros donde los posibles valores de los hiper-parámetros están distribuidos de manera aleatoria a lo largo del espacio finito de hiper-parámetros.

Búsqueda en cuadrícula (Grid Search)

La búsqueda en cuadrícula (Grid Search) es un proceso de búsqueda donde los diferentes valores de hiper-parámetros se combinan para crear una malla (grid) donde se incluyen

todas las posibles combinaciones de parámetros distribuidos de manera uniforme. En ese caso el proceso de búsqueda consiste en utilizar acciones que permiten al algoritmo moverse a través de la cuadrícula seleccionando las mejores selecciones de parámetros en base al resultado obtenido por la función objetivo. En la imagen podemos observar la representación espacial de un proceso de búsqueda en cuadrícula para diferentes hiper-parámetros donde los posibles valores de los hiper-parámetros están distribuidos de manera espacial en una cuadrícula.

Grid Layout



[Bergstra&Bengio \(CC BY-SA\)](#)

Estas técnicas no siempre van a ser aplicables. Por ejemplo, en el caso de redes neuronales, no es posible aplicar este tipo de optimización y ya hay que recurrir a la experiencia u otras funciones de automatización de la búsqueda de buenos hiperparámetros, pero eso se verá en el siguiente módulo.



Autoevaluación

¿Qué técnicas se pueden utilizar en la búsqueda de los hiperparámetros óptimos para nuestro modelo?

- Random Search
- Grid Search
- Las dos anteriores

También valdría Random Search

También sería la de Random Search

Opción correcta

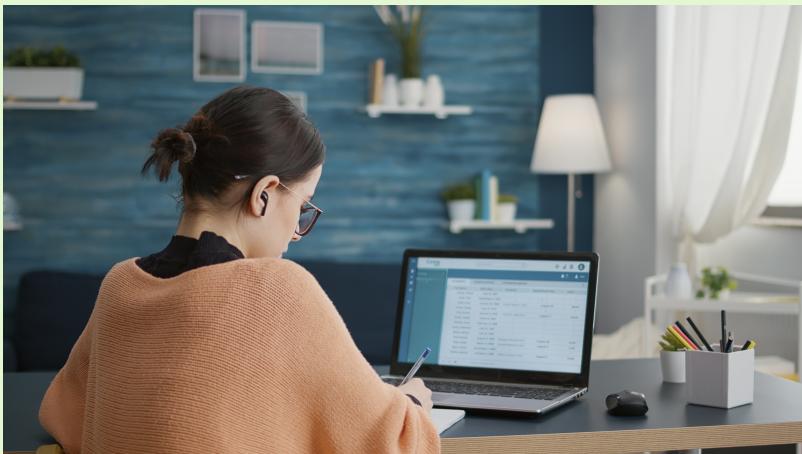
Solución

1. Incorrecto
2. Incorrecto
3. Opción correcta

3.- Datasets no balanceados.



Caso práctico



[DCStudio \(CC BY-SA\)](#)

Max ha estado trabajando con un dataset para un modelo de clasificación en el que hay muy poquitos casos de una clase frente a muchos más de la otra clase, y las predicciones le salen con muchos falsos negativos por ello.

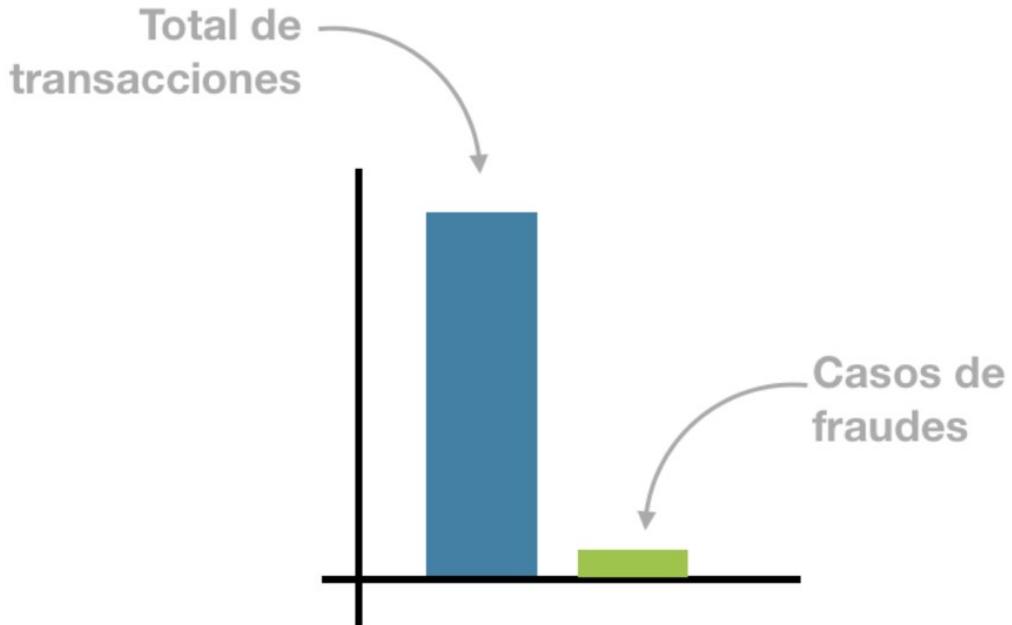
Max recuerda que, en algún momento de la última tutoría, su profesor Antonio le hizo la advertencia "Si tienes un dataset muy desbalanceado tendrás que tratar los datos para evitar que las predicciones salgan distorsionadas".

Por otro lado, Max necesitaría poder estudiar y analizar, de forma aislada esos casos especiales, para buscar la correlación clave entre ellos. Sobre esto, Antonio le recomienda: "Utiliza un algoritmo de reducción dimensional. Hay herramientas que lo hacen automáticamente" y le pasa la referencia de una que Max ya conoce.

"¡Gracias! Miraré el tutorial y lo aplicaré a mi dataset" Contesta Max, contenta.

En los problemas de clasificación en donde tenemos que etiquetar por ejemplo entre "spam" o "not spam" ó entre múltiples categorías (coche, barco, avión) solemos encontrar que en nuestro conjunto de datos de entrenamiento contamos con que alguna de las clases de muestra es una clase "minoritaria" es decir, de la cual tenemos muy pocas muestras. Esto provoca un desbalanceo en los datos que utilizaremos para el entrenamiento de nuestra máquina.

Un caso evidente es en el área de Salud en donde solemos encontrar conjuntos de datos con miles de registros con pacientes "negativos" y unos pocos casos positivos es decir, que padecen la enfermedad que queremos clasificar.



aprendeia.com (Dominio público)

Otros ejemplos suelen ser los de Detección de fraude donde tenemos muchas muestras de clientes “honestos” y pocos casos etiquetados como fraudulentos, o en un funnel de marketing, en donde por lo general tenemos un 2% de los datos de clientes que “compran” ó ejecutan algún tipo de acción (CTA) que queremos predecir.

Por lo general afecta a los algoritmos en su proceso de generalización de la información y perjudicando a las clases minoritarias. Esto suena bastante razonable: si a una red neuronal le damos 990 de fotos de gatitos y sólo 10 de perros, no podemos pretender que logre diferenciar una clase de otra. Lo más probable que la red se limite a responder siempre “tu foto es un gato” puesto que así tuvo un acierto del 99% en su fase de entrenamiento.

Cuando tenemos un dataset con desequilibrio, suele ocurrir que obtenemos un alto valor de precisión en la clase Mayoritaria y un bajo recall en la clase Minoritaria



Autoevaluación

Cuando tenemos un dataset con desequilibrio, suele ocurrir que obtenemos un alto valor de precisión en la clase minoritaria y un bajo recall en la clase mayoritaria

- Verdadero Falso

Falso

Sería justo al revés, suele ocurrir que obtenemos un alto valor de precisión en la clase Mayoritaria y un bajo recall en la clase Minoritaria

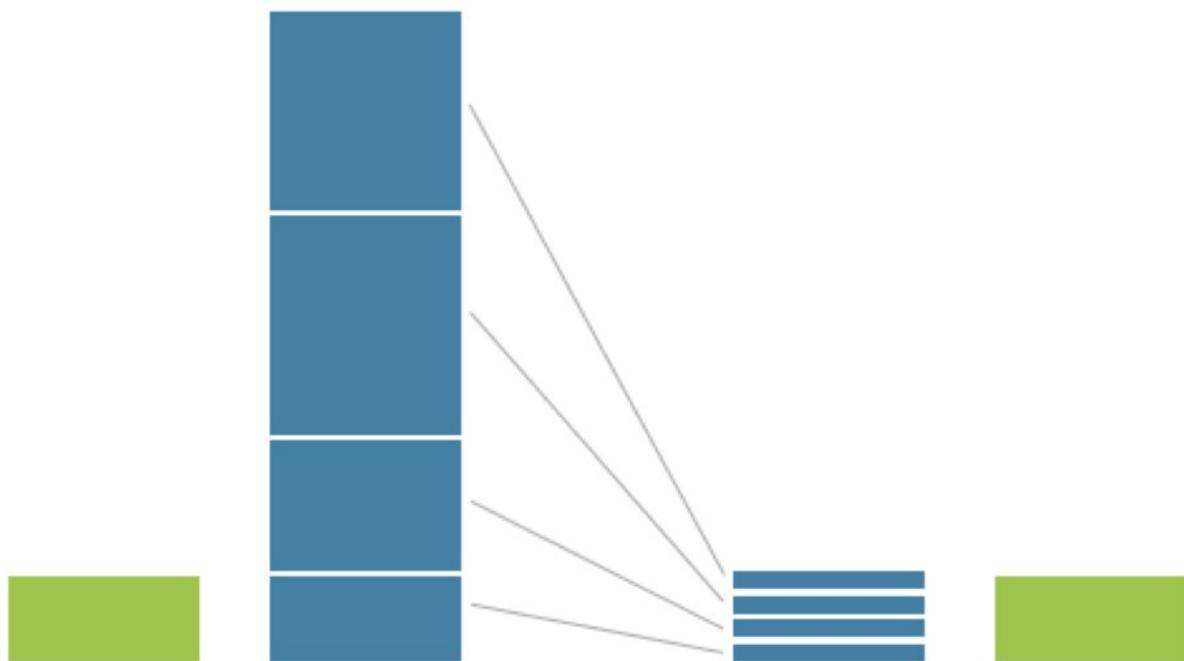
3.1.- Tratamiento de los datos en datasets desbalanceados.

Existen varias técnicas para evitar los problemas que surgen cuando se trabaja con un dataset muy desbalanceado. En el aprendizaje automático, el submuestreo y el sobremuestreo son dos técnicas que se ocupan de los desequilibrios en un conjunto de entrenamiento (la parte de los datos utilizada para ajustar un modelo). Se puede submuestrear la clase mayoritaria, sobremuestrear la clase minoritaria o combinar las dos técnicas.

Es más eficaz, en general, el submuestreo de la clase mayoritaria, especialmente si tenemos un conjunto de datos grande. Pero, como en la mayoría de técnicas y métodos vistos en aprendizaje automático, dependerá mucho de cada caso concreto y siempre es recomendable hacer varias pruebas.

Submuestreo (Under-sampling)

El submuestreo implica la selección de ejemplos de la clase mayoritaria para eliminarlos del conjunto de datos de entrenamiento. Esto tiene el efecto de reducir el número de ejemplos en la clase mayoritaria en la versión transformada del conjunto de datos de entrenamiento. El submuestreo aleatorio , elimina aleatoriamente miembros de la clase mayoritaria hasta alcanzar un umbral preestablecido.

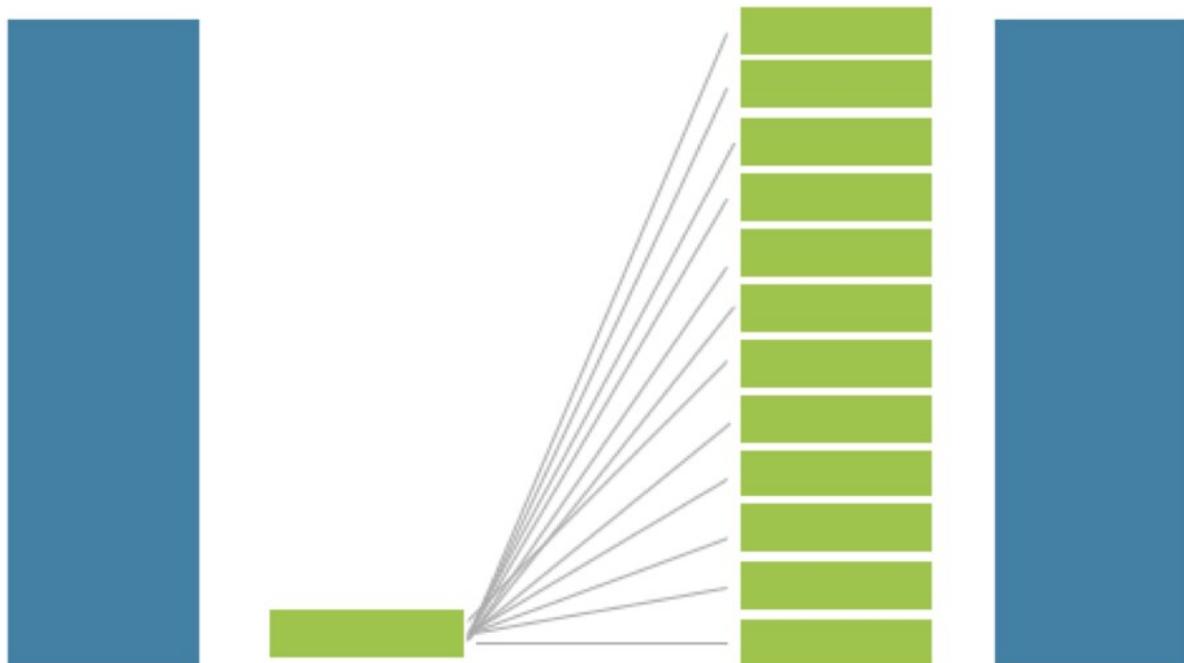


Una ventaja de la selección aleatoria es que no es necesario tomar decisiones sobre qué puntos son importantes y cuáles no: simplemente se automatiza la eliminación de una cierta cantidad de casos. Varios estudios han demostrado que la selección aleatoria funciona tan bien, o incluso mejor, que los procesos en los que se realizan elecciones de eliminación deliberada.

Sin embargo, una clara desventaja es que el proceso podría eliminar miembros importantes. Los problemas tienden a dar como resultado datos que no son homogéneos, tienen límites o características pequeñas.

Sobremuestreo (Over-sampling)

Al igual que el submuestreo, también se puede optar por un sobremuestreo aleatorio. En este caso, se aumentan los casos de la clase minoritaria, de forma sintética, hasta alcanzar una escala similar al volumen de la clase mayoritaria.



aprendeia.com (CC BY-SA)

La ventaja es que no hay riesgo de perder información, pero la desventaja es que este método, con volúmenes de datos muy grandes, penaliza el entrenamiento y aumenta el riesgo de overfitting.

Para este método, se suele utilizar el algoritmo SMOTE.

Otras estrategias

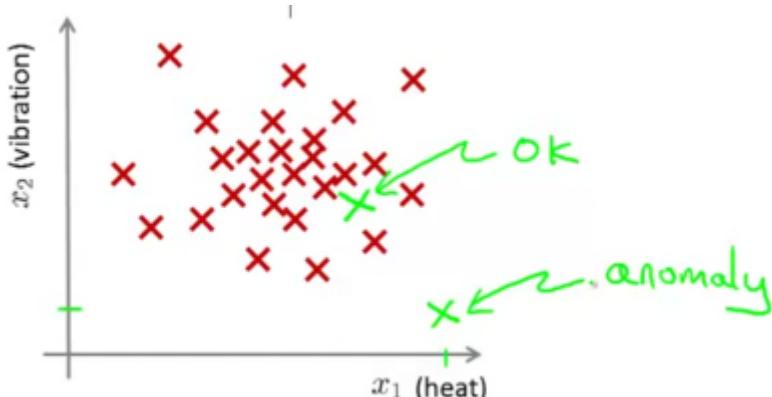
- ✓ Algunos modelos admiten parámetros para contrarrestar el efecto de este desequilibrio, como el peso en árboles de decisión o el parámetro `class_weight` en la regresión logística.
- ✓ Aplicar técnicas "Ensemble" como el algoritmo Random Forest, que entrena un cierto número de modelos y el resultado final se "vota". O también, por ejemplo, en un XGBoost, aumentando el número de árboles, podemos ir corrigiendo los errores de los árboles anteriores.
- ✓ Usar algoritmos de Stacking y algoritmos de aprendizaje por refuerzo: del mismo modo que los Boosting, estos algoritmos permiten ir mejorando los aciertos de la clase minoritaria.

3.2.- Detección de anomalías.

La detección de anomalías (o detección atípica) es la identificación de elementos raros, eventos u observaciones que generan sospechas al diferenciarse significativamente de la mayoría de los datos. Normalmente, los datos anómalos se pueden conectar a algún tipo de problema o evento raro como, por ejemplo, fraude bancario, problemas médicos, defectos estructurales, equipo defectuoso, etc. Esta conexión hace que sea muy interesante poder detectar los casos del conjunto de datos que pueden considerarse anomalías, ya que identificar estos eventos suele ser estratégico desde una perspectiva empresarial.

Pero para que esta situación sea realmente útil, es de vital importancia saber reconocer si estamos ante anomalías debidas a un caso de interés o las debidas a errores de medición o de comunicación.

La industria es uno de los sectores donde este tipo de algoritmos tiene una mayor aplicación. Uno de los usos más comunes es el control de calidad, donde pueden ayudar a reprogramar los ordenadores industriales para la producción de nuevos ítems. También pueden ayudar a optimizar cadenas de suministro detectando donde se producen anomalías en tiempos muertos, así como muchos otros usos como el análisis de comportamiento de compras de clientes, gestión de inventarios, etc.



[Dhananjay Nene](#) (Dominio público)

El ámbito sanitario es uno de los sectores en los que más se está aplicando la inteligencia artificial. Existen muchos trabajos de diagnóstico y monitorización de pacientes que pueden analizarse mediante detección de anomalías, como por ejemplo la detección de planes de tratamiento erróneos en base a series de datos de radioterapia. También tiene un interesante uso en epidemiología, donde permite detectar la aparición de mutaciones de patógenos en base a la respuesta que los pacientes tienen a los tratamientos.

En el sector financiero, uno de los principales usos de la detección de anomalías es descubrir fraudes en pagos electrónicos. También es interesante su aplicación para la detección de solvencia al otorgar créditos, la predicción de bancarrotas o sus diversas aplicaciones para optimizar las inversiones en bolsa.

Reducción de dimensionalidad con PCA y Autoencoders

La detección de anomalías (outliers) con PCA y Autoencoders es una estrategia no supervisada para identificar anomalías cuando los datos no están etiquetados, es decir, no se

conoce la clasificación real (anomalía - no anomalía) de las observaciones.

Si bien esta estrategia hace uso de PCA o Autoencoders, no utiliza directamente su resultado como forma de detectar anomalías, sino que emplea el error de reconstrucción producido al revertir la reducción de dimensionalidad. El error de reconstrucción como estrategia para detectar anomalías se basa en la siguiente idea: los métodos de reducción de dimensionalidad permiten proyectar las observaciones en un espacio de menor dimensión que el espacio original, a la vez que tratan de conservar la mayor información posible. La forma en que consiguen minimizar la perdida global de información es buscando un nuevo espacio en el que la mayoría de observaciones puedan ser bien representadas.

Los métodos de PCA y Autoencoders crean una función que mapea la posición que ocupa cada observación en el espacio original con el que ocupa en el nuevo espacio generado. Este mapeo funciona en ambas direcciones, por lo que también se puede ir desde el nuevo espacio al espacio original. Solo aquellas observaciones que hayan sido bien proyectadas podrán volver a la posición que ocupaban en el espacio original con una precisión elevada.

Dado que la búsqueda de ese nuevo espacio ha sido guiada por la mayoría de las observaciones, serán las observaciones más próximas al promedio las que mejor puedan ser proyectadas y en consecuencia mejor reconstruidas. Las observaciones anómalas, por el contrario, serán mal proyectadas y su reconstrucción será peor. Es este error de reconstrucción (elevado al cuadrado) el que puede emplearse para identificar anomalías.

Isolation Forest

Otro algoritmo típico para la detección de anomalías es el Isolation Forest. La lógica que sigue es diferente a otros métodos conocidos y gira en torno a la idea de que los puntos anormales dentro de los conjuntos de datos son más fáciles de separar (isolate) que los puntos normales. Para lograr esto, el algoritmo genera particiones del conjunto de datos seleccionando un atributo de forma aleatoria, después toma un valor también aleatorio de ese atributo y divide la muestra en dos partes, agrupando los que están por encima y por debajo de dicho valor. Estas operaciones se repiten hasta que todas las observaciones quedan aisladas.

La principal diferenciación de este algoritmo es que requiere menos capacidad de procesamiento que otros métodos, lo que le hace especialmente indicado para grandes conjuntos de datos (datos de alta dimensión).

La herramienta para desarrollo de modelos de aprendizaje automático BigML cuenta con una implementación automatizada del algoritmo Isolation Forest.



[BigML](#) (Dominio público)



Para saber más

Si quieres profundizar más en la detección de anomalías con la herramienta gratuita online BigML, te recomendamos que te descargas la guía "[BigML Anomaly Detection](#)" y practiques con uno de los datasets desbalanceados que vienen precargados en la sección de Fuentes de Datos