

# Curso 1 - Introducción al aprendizaje automático en la producción

En el primer curso de la especialización en ingeniería de aprendizaje automático para la producción, identificará los distintos componentes y diseñará un sistema de producción de ML de principio a fin: alcance del proyecto, necesidades de datos, estrategias de modelado y limitaciones y requisitos de despliegue; y aprenderá a establecer una línea de base del modelo, a abordar la deriva del concepto y a crear un prototipo del proceso para desarrollar, desplegar y mejorar continuamente una aplicación de ML en producción.

Entender los conceptos de aprendizaje automático y aprendizaje profundo es esencial, pero si quieres construir una carrera efectiva en el campo de la IA, también necesitas capacidades de ingeniería de producción. La ingeniería de aprendizaje automático para la producción combina los conceptos fundamentales del aprendizaje automático con la experiencia funcional de los roles modernos de desarrollo de software e ingeniería para ayudarlo a desarrollar habilidades listas para la producción. Semana 1: Visión general del ciclo de vida de ML y su implementación, Semana 2: Selección y entrenamiento de un modelo, Semana 3: Definición de datos y línea de base

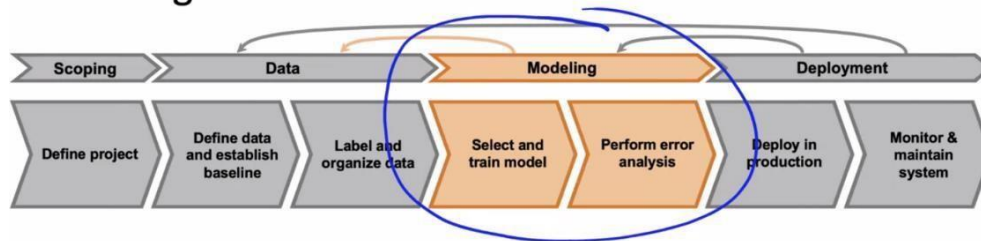
## Semana 2: Seleccionar y entrenar un modelo

### Contenido

<b>Semana 2: Seleccionar y entrenar un modelo</b>	<b>1</b>
Resumen de la modelización	2
Desafíos clave	2
Por qué un error medio bajo no es suficiente	4
Establecer una línea de base	6
Consejos para empezar	9
Ejemplo de análisis de errores	10
Priorizar lo que hay que trabajar	12
Conjuntos de datos sesgados	14
Auditoría de resultados	16
Desarrollo de la IA centrada en los datos	19
Una imagen útil del aumento de datos	20
Aumento de datos	21
¿Puede hacer daño añadir datos?	23
Añadir características	25
Seguimiento del experimento	27
De los big data a los buenos datos	29
Lecturas	29

## Resumen de la modelización

### Modeling



Model-centric AI  
development

Data-centric AI  
development

- Uno de los temas a los que me han oído referirme varias veces es el desarrollo de la IA centrada en los modelos frente al desarrollo de la IA centrada en los datos.
- Tal y como se ha desarrollado la IA, se ha hecho mucho hincapié en cómo elegir el modelo adecuado, por ejemplo, cómo elegir la arquitectura de red neuronal correcta.
- Descubrí que para los proyectos prácticos, puede ser incluso más útil adoptar un enfoque más **centrado en los datos**, en el que uno se centra no sólo en mejorar la arquitectura de la red neuronal, sino en asegurarse de que está alimentando su algoritmo con datos de alta calidad.
- Eso te permite ser más eficiente a la hora de conseguir que tu sistema funcione bien. Pero la forma en que me dedico al desarrollo de la IA centrada en los datos no es simplemente ir a recoger más datos, lo que puede llevar mucho tiempo, sino utilizar herramientas que me ayuden a mejorar los datos de la forma más eficiente posible.

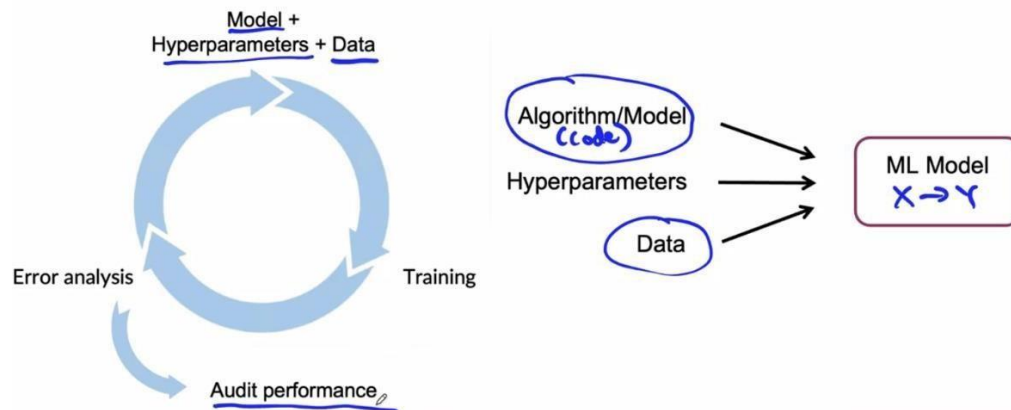
## Desafíos clave

$$\text{AI system} = \text{Code} + \text{Data}$$

(algorithm/model) ↑

- Un marco que espero que tengas en cuenta al desarrollar sistemas de aprendizaje automático es que, los sistemas de IA de los sistemas de aprendizaje automático comprenden tanto el código, es decir, el algoritmo o el modelo, como los datos.
- En las últimas décadas se ha insistido mucho en cómo mejorar el código.
- De hecho, muchas investigaciones implican que los investigadores descarguen conjuntos de datos y traten de encontrar un modelo general que funcione bien en el conjunto de datos.
- Pero para muchas aplicaciones, tienes la flexibilidad de cambiar los datos si no te gustan. Y así, hay muchos proyectos en los que el algoritmo o el modelo es básicamente un problema resuelto.
- Algún modelo que te descargues de GitHub te servirá de sobra, y será más eficiente dedicar gran parte de tu tiempo a mejorar los datos, ya que éstos están mucho más personalizados para tu problema.

## Model development is an iterative process



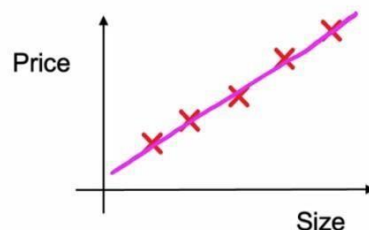
- Entrando en más detalles, cuando se construye un sistema de aprendizaje automático, se puede tener un algoritmo o un modelo, esto sería su código y algunos datos.
- Por supuesto, los hiperparámetros son una entrada adicional a este proceso. Para muchas aplicaciones es importante asegurarse de que se han ajustado las tasas de aprendizaje y los parámetros de regularización, etc.
- Porque el espacio de los hiperparámetros suele ser relativamente limitado, así que voy a dedicar más tiempo a los datos y al código.
- El desarrollo de modelos es un proceso altamente iterativo. Dado que el aprendizaje automático es un proceso tan empírico, ser capaz de pasar por este bucle muchas veces rápidamente es clave para mejorar el rendimiento.
- Pero una de las cosas que te ayudará a mejorar el rendimiento, cada vez que pases por el bucle, es ser capaz de tomar buenas decisiones sobre cómo modificar los datos o cómo modificar el modelo o cómo modificar los hiperparámetros.
- Después de haber hecho esto suficientes veces y haber conseguido un buen modelo, un último paso que suele ser útil es llevar a cabo un análisis de errores más rico y hacer que su sistema pase por una auditoría final para asegurarse de que está funcionando antes de llevarlo a un despliegue de producción.

## Challenges in model development

1. Doing well on training set (usually measured by average training error).

2. Doing well on dev/test sets.

3. Doing well on business metrics/project goals.



- Cuando se construye un modelo, hay **tres hitos clave** que los proyectos deben aspirar a cumplir.
- En primer lugar, debes asegurarte de que lo haces bien en el conjunto de entrenamiento. Así que, si estás prediciendo los precios de la vivienda en función del tamaño de la misma, ¿eres capaz al menos de encajar una línea en tu conjunto de entrenamiento bastante bien?
- Una vez que lo has hecho bien en el conjunto de entrenamiento, tienes que preguntarte si tu algoritmo lo hace bien en el conjunto de desarrollo o en el conjunto de validación cruzada, y finalmente en el conjunto de prueba.

- Si tu algoritmo ni siquiera funciona bien en el conjunto de entrenamiento, es muy poco probable que funcione bien en el conjunto de desarrollo o en el conjunto de prueba.
- Después de hacerlo bien en el conjunto de desarrollo o en el conjunto de pruebas, también tienes que asegurarte de que tu algoritmo de aprendizaje lo hace bien según las métricas del negocio o según los objetivos del proyecto.
- En las últimas décadas, gran parte del desarrollo del aprendizaje automático ha estado impulsado por el objetivo de obtener buenos resultados en el conjunto de desarrollo o en el conjunto de pruebas.
- Lamentablemente, para muchos problemas, tener una alta precisión probada no es suficiente para alcanzar los objetivos del proyecto. Y esto ha provocado mucha frustración y desacuerdos entre el equipo de ejecución de la máquina, que es muy bueno en esto, y los equipos de negocio que se preocupan más por las métricas de negocio o algunos otros objetivos del proyecto.

## Por qué un error medio bajo no es suficiente

### Performance on disproportionately important examples



#### Web Search example

"Apple pie recipe"

"Latest movies"

"Wireless data plan"

"Diwali festival"

**Informational and  
Transactional queries**

"Stanford"

"Reddit"

"Youtube"

**Navigational queries**

- Un sistema de aprendizaje automático puede tener un error medio bajo en el conjunto de pruebas, pero si su rendimiento en un conjunto de ejemplos desproporcionadamente importantes no es lo suficientemente bueno, el sistema de aprendizaje automático seguirá sin ser aceptable para la implantación en producción.
- Permítanme utilizar un ejemplo de búsqueda en la web. Hay muchas consultas de búsqueda en la web como éstas: Receta de tarta de manzana, últimas películas y plan de datos inalámbrico.
- Este tipo de consultas se denominan a veces **consultas informativas o transaccionales**, en las que quiero aprender sobre las tartas de manzana, o tal vez quiero comprar un nuevo plan de datos inalámbricos
- Para las consultas informativas y transaccionales, puede estar dispuesto a perdonar a un motor de búsqueda que no le dé la mejor receta de tarta de manzana, porque hay muchas buenas recetas de tarta de manzana en Internet.
- Hay un tipo diferente de consultas de búsqueda en la web, denominadas **consultas de navegación**, en las que el usuario tiene una intención muy clara, un deseo muy claro de ir a Stanford.edu, o a Reddit.com, o a YouTube.com.
- Cuando un usuario tiene una intención de navegación muy clara, tenderá a ser muy implacable si un motor de búsqueda web hace cualquier cosa que no sea devolver Stanford.edu como el resultado número uno.
- El motor de búsqueda que no ofrezca resultados correctos perderá rápidamente la confianza de sus usuarios.
- Las consultas de navegación en este contexto son un conjunto de ejemplos desproporcionadamente importante y si tienes un algoritmo de aprendizaje que mejora la precisión media del conjunto de pruebas para la búsqueda en la web pero estropea sólo un pequeño puñado de consultas de navegación, puede que no sea aceptable para su despliegue.
- El reto, por supuesto, es que la precisión media del conjunto de pruebas tiende a ponderar todos los ejemplos por igual, mientras que en la búsqueda web, algunas consultas son desproporcionadamente importantes.

- Ahora bien, una cosa que se podría hacer es intentar dar a estos ejemplos un peso mayor. Eso podría funcionar para algunas aplicaciones, pero en mi experiencia, el **simple hecho de cambiar los pesos de los diferentes ejemplos no siempre resuelve todo el problema**

## Performance on key slices of the dataset

### Example: ML for loan approval

Make sure not to discriminate by ethnicity, gender, location, language or other protected attributes.

### Example: Product recommendations from retailers

Be careful to treat fairly all major user, retailer, and product categories.

- Digamos que ha construido un algoritmo de aprendizaje automático para la aprobación de préstamos para decidir quién es probable que devuelva un préstamo, y así recomendar la aprobación de ciertos préstamos para su aprobación.
- Para un sistema de este tipo, probablemente querrá asegurarse de que su sistema no discrimina injustamente a los solicitantes de préstamos en función de su etnia, género, tal vez su ubicación, su idioma u otros atributos protegidos.
- Incluso si un algoritmo de aprendizaje para la aprobación de préstamos alcanza una alta precisión media en el conjunto de pruebas, no sería aceptable para su despliegue en producción si muestra un nivel inaceptable de sesgo o discriminación.
- La comunidad de la Inteligencia Artificial ha debatido mucho sobre la equidad con las personas, y con razón, porque es un tema importante que tenemos que abordar y hacer bien.
- Supongamos que usted dirige un sitio web de compras en línea en el que defiende y vende productos de muchos fabricantes diferentes y de muchas marcas distintas de minoristas.
- Es posible que quiera asegurarse de que su sistema trata de forma equitativa todas las categorías principales de usuarios, minoristas y productos. Por ejemplo, aunque un sistema de aprendizaje automático tenga una alta precisión media en el conjunto de pruebas, puede que recomiende mejores productos de media. Si da recomendaciones realmente irrelevantes a todos los usuarios de una etnia, eso puede ser inaceptable
- O si siempre recomienda los productos de los grandes distribuidores e ignora las marcas más pequeñas, también podría ser perjudicial para el negocio, ya que podría perder a todos los pequeños distribuidores. También sería injusto crear un sistema de recomendación que sólo recomendara productos de las grandes marcas e ignorara a las pequeñas empresas.

## Rare classes

Skewed data distribution

99% negative 1% positive

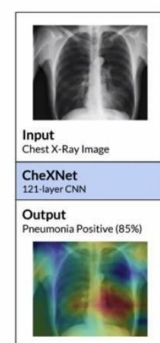
print("0") ←

10,000 →

~100 →

Accuracy in rare classes

Condition	Performance
Effusion	0.901 ←
Edema	0.924
Mass	0.909
<u>Hernia</u>	0.851 ←



- A continuación está el tema de las clases raras, y en concreto el de las **distribuciones de datos sesgadas**.
- En el diagnóstico médico, no es raro que muchos pacientes no tengan una determinada enfermedad,

- Si se tiene un conjunto de datos con un 99% de ejemplos negativos, puede ser porque el 99% de la población no tiene una determinada enfermedad, y sólo el 1% es positivo.
- Se puede lograr una muy buena precisión probada con sólo escribir un programa que simplemente diga imprimir "0", sin tener que entrenar una enorme red neuronal que haga lo mismo
- La **precisión** en las clases raras está estrechamente relacionada con la cuestión de las distribuciones de datos sesgadas.
- Estaba trabajando en el diagnóstico de radiografías de tórax, y utilizando modelos de aprendizaje profundo para detectar diferentes condiciones
- La efusión es una condición médica común, en la que teníamos alrededor de 10.000 imágenes y por lo tanto, fuimos capaces de lograr un alto nivel de rendimiento
- Sin embargo, para la afección mucho más rara de la hernia, sólo teníamos un centenar de imágenes, por lo que el rendimiento era mucho peor.
- Resulta que, desde el punto de vista médico, no es aceptable que el sistema de diagnóstico ignore los casos evidentes de hernia. Al tratarse de una clase relativamente rara, la precisión media global del algoritmo no fue tan mala
- De hecho, el algoritmo podría haber ignorado por completo todos los casos de hernia y sólo habría tenido un impacto modesto en esta precisión media de la prueba
- Dado que los casos de hernia son poco frecuentes, el algoritmo podría ignorarlos sin perjudicar demasiado la precisión media de la prueba, especialmente si la precisión media del conjunto de pruebas da la misma importancia a cada uno de los ejemplos del conjunto de pruebas.

## Unfortunate conversation in many companies



MLE: "I did well on the test set!"



Product Owner: "But this doesn't work for my application"



MLE: "But... I did well on the test set!"

- He escuchado exactamente esta misma conversación demasiadas veces en demasiadas empresas y la conversación va así, un ingeniero de aprendizaje automático dice,
- "¡Lo hice bien en el conjunto de pruebas!" esto funciona, y el dueño del negocio dice, "pero esto no funciona para mi aplicación" y el ingeniero de aprendizaje automático responde, "¡pero lo hice bien en el conjunto de pruebas!"
- Si alguna vez te encuentras en esta conversación, no te pongas a la defensiva.
- Como comunidad, hemos creado muchas herramientas para obtener buenos resultados en el conjunto de pruebas, y eso hay que celebrarlo. Pero a menudo tenemos que ir más allá, porque el mero hecho de hacerlo bien en el conjunto de pruebas
- Hacerlo bien en el conjunto de pruebas no es suficiente para muchas aplicaciones de producción.
- Cuando construyo un sistema de aprendizaje automático, considero que mi trabajo no es sólo hacerlo bien en el conjunto de pruebas, sino producir un sistema de aprendizaje automático que **resuelva las necesidades reales del negocio o de la aplicación**, y espero que tú también tengas una visión similar.

## Establecer una línea de base

- Cuando se empieza a trabajar en un proyecto de aprendizaje automático, uno de los primeros pasos más útiles es establecer una línea de base. Por lo general, solo después de haber establecido un nivel de referencia de rendimiento se puede disponer de herramientas para mejorar eficazmente ese nivel de referencia.



# Establishing a baseline level of performance





## Speech recognition example:

Type	Accuracy	Human level performance	
Clear Speech	94%	95%	10%
Car Noise	89%	93%	4%
People Noise	87%	89%	2%
→ <u>Low Bandwidth</u>	<u>70%</u>	<u>70%</u>	<u>~0%</u>

- Digamos que ha establecido que hay cuatro categorías principales de discurso en sus datos.
- Con su precisión en estas cuatro categorías de discursos como 94, 89, 87 y 70 por ciento de precisión respectivamente, podría estar tentado a centrar nuestra atención en el audio de bajo ancho de banda (menor % de precisión)
- Pero antes de llegar a esa conclusión, sería útil establecer un nivel de referencia de rendimiento en las cuatro categorías. Para ello, puedes pedir a algunos transcripores humanos que etiqueten tus datos y medir su precisión.
- Esto determina cuál es el nivel de rendimiento humano (HLP) en estas cuatro categorías de discurso
- En este ejemplo, encontramos que si podemos mejorar nuestro rendimiento en el habla clara hasta el nivel de rendimiento humano, parece que hay un potencial de mejora del uno por ciento allí. Si podemos aumentar nuestro rendimiento hasta el nivel humano en el audio con ruido de coche de fondo, sería una mejora del cuatro por ciento, y una mejora del cero por ciento en el audio de bajo ancho de banda.
- Con este análisis, nos dimos cuenta de que el audio de bajo ancho de banda era tan confuso que ni siquiera los humanos pueden reconocer lo que se decía. Por lo tanto, puede que no sea tan fructífero trabajar en ello.
- En cambio, puede ser más fructífero centrar nuestra atención en mejorar el reconocimiento del habla con el ruido del coche de fondo.
- El uso de HLP le proporciona un punto de comparación o una línea de base que le ayuda a decidir dónde centrar sus esfuerzos en los datos de ruido del coche en lugar de en los datos de bajo ancho de banda.

## Unstructured and structured data

Unstructured data		Structured data			
Image		User ID	Purchase	Number	Price
Audio		3421	Blue shirt	5	\$20
Text	<div>This restaurant was great!</div>	612	Brown shoes	1	\$35
HLP		Product ID	Product name	Inventory	
		385	Football	158	
		477	Cricket bat	23	

- Resulta que las mejores prácticas para establecer una línea de base son bastante diferentes, dependiendo de si se trabaja con datos no estructurados o estructurados.

- Los datos no estructurados se refieren a conjuntos de datos como imágenes, tal vez fotos de gatos o audio, como nuestro ejemplo de reconocimiento de voz o lenguaje natural
- **Los datos no estructurados** tienden a ser datos que **los humanos son muy buenos** para interpretar (por ejemplo, entender imágenes y audio)
- Dado que los seres humanos son tan buenos en las tareas de datos no estructurados, la medición del rendimiento a nivel humano o HLP, suele ser una buena forma de establecer una línea de base si se trabaja con datos no estructurados.
- En cambio, los datos estructurados son las gigantescas bases de datos que puede tener, por ejemplo, si gestiona un sitio web de comercio electrónico, con los datos que muestran qué usuarios compraron en qué momento y a qué precio.  
Los seres humanos no son tan buenos mirando datos como estos para hacer predicciones. Ciertamente no evolucionamos para mirar hojas de cálculo gigantes.
- El rendimiento a nivel humano suele ser un punto de referencia menos útil para las aplicaciones de datos estructurados.
- Las mejores prácticas de desarrollo del aprendizaje automático son bastante diferentes, dependiendo de si se trabaja con datos no estructurados o con problemas de datos estructurados.
- Teniendo en cuenta esta diferencia, veamos algunas formas de establecer líneas de base para ambos tipos de problemas

## Ways to establish a baseline

- Human level performance (HLP)
- Literature search for state-of-the-art/open source
- Quick-and-dirty implementation
- Performance of older system

Baseline helps to indicates what might be possible. In some cases (such as HLP) is also gives a sense of what is irreducible error/Bayes error.

- Otra forma de establecer una línea de base es hacer una **búsqueda bibliográfica del estado del arte**, o mirar los resultados de código abierto para ver lo que otros informan
- Por ejemplo, si estás construyendo un sistema de reconocimiento de voz y otros informan de un cierto nivel de precisión en datos similares a los tuyos, eso puede darte un punto de partida.
- Al utilizar el código abierto, también puede considerar la posibilidad de realizar una implementación rápida y sucia, que podría empezar a darle una idea de lo que puede ser posible.



- Por último, si ya tiene un sistema de aprendizaje automático en funcionamiento para su aplicación, el rendimiento de su sistema anterior, el rendimiento de su sistema más antiguo también puede ayudarlo a establecer una línea de base que luego puede aspirar a mejorar.
- En algunos casos, como cuando se utiliza el rendimiento a nivel humano, especialmente en problemas de datos no estructurados, esta línea de base también puede dar una idea de cuál es el error irreducible o cuál es el error de Bayes. En otras palabras, ¿qué es lo mejor que se puede esperar en términos de rendimiento en este problema?
- Por ejemplo, podemos darnos cuenta de que el audio de bajo ancho de banda es tan malo que no es posible tener más del 70% de precisión
- He visto a equipos empresariales presionar a un equipo de aprendizaje automático para que garantice que el algoritmo tendrá una precisión del 80% o incluso del 99%, antes de que el equipo haya tenido siquiera la oportunidad de establecer una línea de base aproximada
- Esto, por desgracia, pone al equipo de aprendizaje automático en una posición muy difícil. Si se encuentra en esa situación, le insto a que considere la posibilidad de dar marcha atrás y **pedir tiempo para establecer un nivel de referencia aproximado de rendimiento** antes de dar una predicción más firme sobre la precisión que puede llegar a tener el sistema de aprendizaje automático.

## Consejos para empezar

### Getting started on modeling

- Literature search to see what's possible (courses, blogs, open-source projects).
  - Find open-source implementations if available.
  - A reasonable algorithm with **good data** will often outperform a great algorithm with no so good data.
- 
- Para empezar a dar este primer paso de la llegada del modelo, he aquí algunas sugerencias.
  - Cuando empiezo a trabajar en un proyecto de aprendizaje automático, casi siempre empiezo con una rápida búsqueda bibliográfica para ver qué es posible, así que puedes mirar cursos online, mirar blogs y mirar proyectos de código abierto.
  - Mi consejo, si tu objetivo es construir un sistema de **producción práctico** y no hacer investigación, es que no te obsesiones con encontrar el último y mejor algoritmo.
  - En su lugar, dedica medio día, o tal vez un pequeño número de días, a leer publicaciones en blogs y elige algo razonable que te permita empezar rápidamente. Si puedes encontrar una implementación de código abierto, eso también puede ayudarte a establecer una línea de base de manera más eficiente.
  - Para muchas aplicaciones prácticas, **un algoritmo razonable con buenos datos suele funcionar bien y, de hecho, superará a un gran algoritmo con datos no tan buenos.**
  - No te obsesiones con coger el algoritmo que se acaba de publicar en alguna conferencia la semana pasada, que es el más puntero. En lugar de eso, busca algo razonable, encuentra una buena implementación de código abierto y úsala para ponerte en marcha rápidamente.
  - Porque ser capaz de empezar en este primer paso de este bucle, puede hacer que sea más eficiente en la iteración a través de más veces, y que le ayudará a llegar a un buen rendimiento más rápidamente.

### Deployment constraints when picking a model

Should you take into account deployment constraints when picking a model?

**Yes**, if baseline is already established and goal is to build and deploy.

**No** (or not necessarily), if purpose is to establish a baseline and determine what is possible and might be worth pursuing.

- Si la línea de base ya está establecida, y estás relativamente seguro de que este proyecto funcionará (es decir, el objetivo es construir y desplegar un sistema), entonces sí, debes tener en cuenta las restricciones de despliegue, como las restricciones de computación
- **Pero** si todavía no ha establecido una línea de base, o si está en una etapa del proyecto en la que su objetivo es sólo establecer una línea de base y determinar si este proyecto vale la pena a largo plazo, entonces podría estar bien ignorar las restricciones de despliegue, y simplemente encontrar alguna implementación de código abierto y probarla para ver lo que podría ser posible.
- Esto es así incluso si esa implementación de código abierto es tan intensiva desde el punto de vista computacional que sabes que nunca serás capaz de desplegarla. Está bien centrarse en establecer eficientemente la línea de base primero.

## Sanity-check for code and algorithm

- Try to overfit a small training dataset before training on a large one.

- Example #1: Speech recognition



- Example #2: Image segmentation



- Example #3: Image classification

10,000  
10, 100

- Al probar un algoritmo de aprendizaje por primera vez, antes de ejecutarlo en todos sus datos,
- Le insto a que realice unas cuantas comprobaciones rápidas de cordura para su código y su algoritmo.
- Por ejemplo, suelo intentar sobreajustar un conjunto de datos de entrenamiento muy pequeño antes de dedicar horas o incluso días a entrenar el algoritmo en un conjunto de datos grande.
- Por ejemplo, una vez trabajé en un sistema de reconocimiento de voz cuyo objetivo era introducir audio y hacer que un algoritmo de aprendizaje produjera una transcripción. Cuando entrené a mi algoritmo con un solo ejemplo, un clip de audio, éste produjo múltiples espacios (blancos)
- Estaba claro que no funcionaba y, como mi sistema de voz ni siquiera podía transcribir con precisión un ejemplo de entrenamiento, no tenía mucho sentido pasar horas entrenándolo con un conjunto de entrenamiento gigante.
- En el caso de la segmentación de imágenes, si el objetivo es tomar como entrada fotografías como ésta y segmentar los gatos de la imagen, antes de pasar horas entrenando el sistema con cientos o miles de imágenes, una buena comprobación de cordura sería alimentarlo con una sola imagen y ver si puede al menos sobreajustar ese ejemplo de entrenamiento, antes de ampliarlo a un conjunto de datos mayor.
- La ventaja de esto es que puedes entrenar tu algoritmo en uno o un pequeño puñado de ejemplos en sólo minutos o incluso segundos y esto te permite encontrar errores mucho más rápidamente.
- Por último, para los problemas de clasificación de imágenes, puede valer la pena entrenar rápidamente su algoritmo en un pequeño subconjunto de sólo 10 o tal vez 100 imágenes, porque puede hacerlo rápidamente. Si tu algoritmo no puede hacerlo bien ni siquiera con 100 imágenes, está claro que no lo hará con 10.000.

### Ejemplo de análisis de errores

- La primera vez que se entrena un algoritmo de aprendizaje, casi se puede garantizar que no funcionará a la primera. Así que creo que el corazón del proceso de desarrollo del aprendizaje automático es el análisis de errores

## Speech recognition example

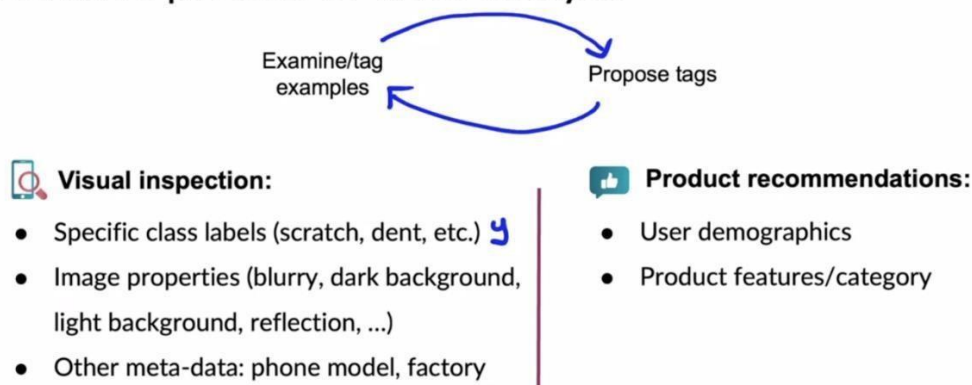
Example	Label	Prediction	Car noise	People noise	Low bandwidth
1	"Stir fried lettuce recipe"	"Stir fry lettuce recipe"	1		
2	"Sweetened coffee"	"Swedish coffee"		1	1
3	"Sail away song"	"Sell away some"		1	
4	"Let's catch up"	"Let's ketchup"	1	1	1

- Cuando llevo a cabo el análisis de errores, esto es más o menos lo que yo mismo haría en una hoja de cálculo para conocer los errores del sistema de voz.
- Es posible que escuches tal vez 100 ejemplos mal etiquetados de tu set de desarrollo.
- Cuando escuches este ejemplo, si tiene ruido de coche de fondo, puedes hacer una anotación en tu hoja de cálculo para indicar que este ejemplo tenía ruido de coche.

Literalmente, disparo un programa de hoja de cálculo como Google sheet o Excel o en un Mac, y lo hago así en la hoja de cálculo.

- Este proceso le ayuda a comprender si las categorías, indicadas por las etiquetas que pueden ser la fuente de más errores y pueden merecer más esfuerzo y atención.
- Hasta ahora, el análisis de errores se ha realizado normalmente mediante un proceso manual, por ejemplo, en el cuaderno Jupyter o rastreando los errores en una hoja de cálculo.
- Yo todavía lo hago así a veces y si tú también lo haces así, está bien. Pero también hay herramientas emergentes de MLOps que facilitan este proceso a los desarrolladores.
- Por ejemplo, cuando mi equipo en LandingAI trabaja en aplicaciones de visión por ordenador, todo el equipo utiliza Landing Lens, lo que facilita mucho las cosas en comparación con la hoja de cálculo.

## Iterative process of error analysis



El entrenamiento de un modelo y su despliegue son procesos iterativos. Así que no debería sorprender que el

- análisis de errores sea también un proceso iterativo
- Un proceso típico es examinar y etiquetar un conjunto de ejemplos con un conjunto inicial de etiquetas, como el ruido de los coches y el ruido de las personas. Y sobre la base de este conjunto inicial de ejemplos, puede volver y decir que quiere proponer algunas etiquetas nuevas.
- Con las nuevas etiquetas, puede volver a examinar y etiquetar aún más ejemplos.
- Tomemos como ejemplo la inspección visual. Ya sabes, el problema de encontrar defectos en los teléfonos inteligentes. Algunas de las etiquetas podrían ser **etiquetas de clase** específicas, como arañazos o abolladuras. Algunas de las etiquetas podrían ser **propiedades de la imagen**, como si esta foto del teléfono está borrosa. ¿Está sobre un fondo oscuro o sobre un fondo claro?
- Las etiquetas también pueden proceder de otras formas de **metadatos**. Por ejemplo, ¿cuál es el modelo de película? ¿Cuál es la fábrica que ha capturado la imagen específica?

- Y el objetivo de este tipo de proceso es tratar de llegar a unas cuantas categorías en las que
- podría mejorar productivamente el algoritmo
- Otro ejemplo son las recomendaciones de productos para un sitio de comercio electrónico en línea. Se podría observar qué productos recomienda un sistema a los usuarios y encontrar las recomendaciones claramente incorrectas o irrelevantes, y tratar de averiguar si hay datos demográficos específicos de los usuarios
- Por ejemplo, ¿realmente recomendamos mal los productos a las mujeres jóvenes o a los hombres mayores o a otra cosa? O hay características específicas de productos o categorías específicas de productos en las que las recomendaciones son especialmente malas.
- Si se hace una lluvia de ideas y se aplican estas etiquetas, se pueden obtener algunas ideas para las categorías de datos en las que se intenta mejorar el rendimiento del algoritmo.

pág. 11

- A medida que vaya revisando estas diferentes etiquetas, aquí hay algunos números útiles para mirar. En primer lugar, ¿qué fracción de errores tiene esa etiqueta? Por ejemplo, si escuchas 100 clips de audio y descubres que el 12% de ellos estaban etiquetados con el tipo de ruido del coche, eso te da una idea de lo importante que es trabajar con el ruido del coche.
- También te dice que incluso si arreglas todos los problemas de ruido del coche, el rendimiento puede mejorar sólo un 12%, lo que en realidad no está mal.
- O puede preguntar a todos los datos con esa etiqueta qué fracción está mal clasificada. Puedes centrar tu atención en etiquetar los ejemplos mal clasificados.
- Se puede preguntar de todos los datos de esa etiqueta qué fracción está mal clasificada. Así, por ejemplo, si encuentras que de todos los datos con ruido de coche, el 18% está mal transcrito. Eso te dice que el rendimiento de los datos con este tipo de etiqueta sólo tiene un cierto nivel de precisión y te dice lo difíciles que son realmente estos ejemplos con ruido de coche.
- También puede preguntar qué fracción de todos los datos tiene esa etiqueta. Esto le indica la importancia que tienen los ejemplos con esa etiqueta en relación con todo el conjunto de datos.
- Por último, ¿cuál es el margen de mejora de los datos con esa etiqueta? Un ejemplo que ya has visto es medir el rendimiento a nivel humano de los datos con esa etiqueta.
- Mediante una lluvia de ideas de diferentes etiquetas, puedes segmentar tus datos en diferentes categorías, y luego utilizar preguntas como estas para tratar de decidir a qué dar prioridad para trabajar.

## Priorizar lo que hay que trabajar

### Useful metrics for each tag

- What fraction of errors has that tag?  $12\%$
- Of all data with that tag, what fraction is misclassified?  $18\%$
- What fraction of all the data has that tag?
- How much room for improvement is there on data with that tag?

### Prioritizing what to work on

Type	Accuracy	Human level performance	Gap to HLP	% of data
Clean Speech	94%	95%	1%	$60\% \rightarrow 0.6\%$
Car Noise	89%	93%	4%	$4\% \rightarrow 0.16\%$
People Noise	87%	89%	2%	$30\% \rightarrow 0.6\%$
Low Bandwidth	70%	70%	0%	$6\% \rightarrow \sim 0\%$

- En lugar de decidir trabajar con el ruido de los coches porque la diferencia con HLP es mayor, otro factor útil que hay que mirar es el **porcentaje de datos con esa etiqueta**.
- Digamos que el 60 por ciento de los datos es habla limpia, el cuatro por ciento son datos con ruido de coches, el 30 por ciento tiene ruido de personas y el seis por ciento es audio de bajo ancho de banda.
- Esto nos dice que si pudiéramos tomar el habla limpia y aumentar nuestra precisión del 94-95 por ciento en todo el habla limpia, entonces multiplicar el uno por ciento con el 60 por ciento sólo nos dice que nuestro **sistema de habla general sería un 0,6 por ciento más preciso**
- En cuanto al ruido de los coches, si podemos mejorar el rendimiento en un cuatro por ciento de los datos, multiplicando eso, nos da una mejora del 0,16 por ciento



Mientras que antes habíamos dicho que había mucho margen de mejora en el ruido de los coches, en este análisis ligeramente más rico, vemos que como el ruido de las personas y el habla limpia representan una fracción tan grande de los datos, puede merecer más la pena trabajar en cualquiera de ellos, porque en realidad hay un mayor potencial de mejora en ambos que en el habla con ruido de los coches.

## Prioritizing what to work on

Decide on most important categories to work on based on:

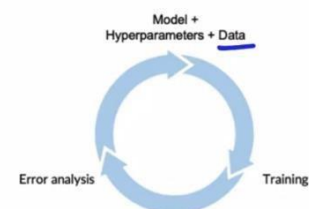
- How much room for improvement there is.
  - How frequently that category appears.
  - How easy is to improve accuracy in that category.
  - How important it is to improve in that category.
- Para resumir, cuando se prioriza en qué trabajar, se puede decidir sobre las categorías más importantes en las que trabajar basándose en el margen de mejora que hay, por ejemplo, en comparación con el rendimiento a nivel humano o según alguna comparación de referencia.
  - ¿Con qué frecuencia aparece esa categoría? También puede tener en cuenta lo fácil que es mejorar la precisión en esa categoría.
  - Por último, decide lo importante que es mejorar el rendimiento en esa categoría. Por ejemplo, puedes decidir que mejorar el rendimiento con el ruido del coche es especialmente importante, porque cuando estás conduciendo, tienes más ganas de hacer búsquedas, como buscar en mapas y encontrar direcciones sin necesidad de usar las manos cuando tienes el volante.
  - No hay una fórmula matemática que te diga en qué debes trabajar. Pero, si observas estos factores, espero que seas capaz de tomar decisiones más fructíferas.

## Adding/improving data for specific categories

For categories you want to prioritize:

- Collect more data
- Use data augmentation to get more data
- Improve label accuracy/data quality

Type	Accuracy	Human level performance	Gap to HLP	% of data
Clean Speech	94%	95%	1%	60%
→ Car Noise	84%	93%	4%	4%
→ People Noise	87%	89%	2%	30%
→ Low Bandwidth	70%	70%	0%	6%



- Una vez que haya decidido que hay una categoría, o tal vez unas cuantas categorías en las que desea mejorar el rendimiento medio, un enfoque fructífero es considerar la posibilidad de añadir datos o mejorar la calidad de los mismos para ese pequeño puñado de categorías.
- Por ejemplo, si quieres mejorar el rendimiento en el habla con ruido de coches, puedes salir a recoger más datos con ruido de coches. O si tienes una forma de utilizar el aumento de datos para obtener más datos de la categoría de datos, esa será otra forma de mejorar el rendimiento de tu algoritmo. En el aprendizaje automático, siempre nos gustaría tener más datos, pero salir a recoger más datos de forma genérica, puede ser muy lento y caro. Llevando a cabo un análisis como este, puedes **centrarte mucho más** en qué tipos de datos necesitas recopilar exactamente para mejorar el rendimiento de tu algoritmo de aprendizaje.

- Porque si se decide recoger más datos con el ruido de los coches o quizás con el ruido de la gente, se puede ser mucho más específico al salir a recoger más datos sólo de ese tipo o al utilizar el aumento de datos, sin perder el tiempo tratando de recoger más datos de una conexión de teléfono móvil de bajo ancho de banda.

## Conjuntos de datos sesgados

### Examples of skewed datasets



#### Manufacturing example

99.7% no defect  $y=0$   
0.3% defect  $y=1$

`print("0")`  
99.7%



#### Medical Diagnosis example: 99% of patients don't have a disease



#### Speech Recognition example: In wake word detection, 96.7% of the time wake word doesn't occur

- Dada una empresa de fabricación de teléfonos inteligentes: Si el 99,7 por ciento no tiene ningún defecto (etiquetado y igual a 0) y sólo una pequeña fracción está etiquetado y igual a 1, entonces imprimir 0 todo el tiempo seguirá dando una precisión del 99,7%, lo que no es un algoritmo de aprendizaje muy impresionante.
- En el caso del diagnóstico médico, si el 99% de los pacientes no tienen una enfermedad, un algoritmo que predice que nadie tiene una enfermedad tendrá una precisión del 99%.
- En el caso del reconocimiento del habla, si estás construyendo un sistema para la detección de palabras de despertador, a veces también llamada detección de palabras de activación, la mayoría de las veces esa palabra especial de despertador o de activación no está siendo pronunciada por nadie en ese momento.
- Cuando construí los sistemas de detección de palabras de estela, los conjuntos de datos estaban bastante sesgados. Uno de los conjuntos de datos que utilicé tenía un 96,7% de ejemplos negativos y un 3,3% de ejemplos positivos.
- Cuando se tiene un conjunto de datos sesgados como éste, la precisión no es una métrica tan útil, ya que imprimir el cero puede conseguir una precisión muy alta.

### Confusion matrix: Precision and Recall

		Actual	
		$y=0$	$y=1$
Predicted	$y=0$	905 TN	18 FN
	$y=1$	9 FP	68 TP
		914	86

TN: True Negative

TP: True Positive

FN: False Negative

FP: False Positive

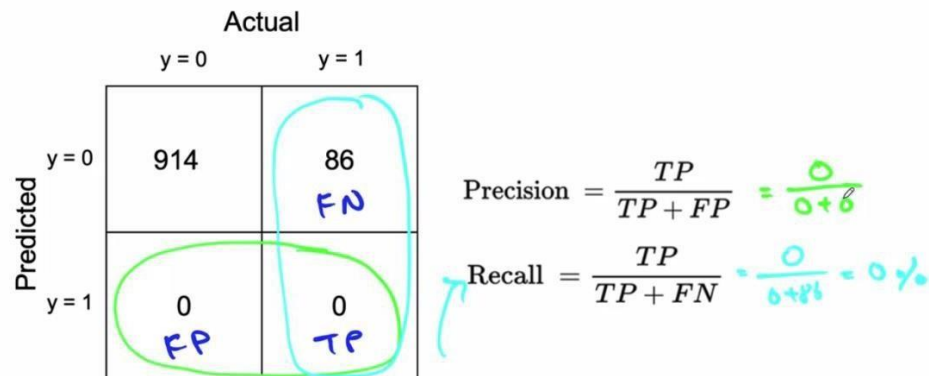
$$\text{Precision} = \frac{TP}{TP+FP} = \frac{68}{68+9} = 88.3\%$$

$$\text{Recall} = \frac{TP}{TP+FN} = \frac{68}{68+18} = 79.1\%$$

En cambio, es más útil construir algo llamado matriz de confusión. Una matriz de confusión es una matriz en la que uno de los ejes está etiquetado con la etiqueta real, es la etiqueta de la verdad sobre el terreno, y es igual a 0 o y es igual a 1, y cuyo otro eje está etiquetado con la predicción.

- Aquí tenemos un conjunto de datos bastante sesgado en el que, de 1.000 ejemplos, había 940 ejemplos negativos y sólo 86 positivos, es decir, un 8,6% positivos y un 91,4% negativos.
- Las métricas de precisión y recuperación son más útiles que la exactitud bruta cuando se trata de evaluar el rendimiento de los algoritmos de aprendizaje en conjuntos de datos sesgados.

## What happens with print("0")?



- Tomando este ejemplo en el que tenemos 914 ejemplos negativos y 86 positivos, si el algoritmo da como resultado cero todo el tiempo, así es como se verá la matriz de confusión
- El algoritmo 0 alcanza un cero por ciento de recuperación, lo que le ofrece una forma fácil de indicar que no está detectando ningún ejemplo positivo útil.
- El algoritmo de aprendizaje, incluso con un valor alto de precisión, no suele ser tan útil si el recuerdo es tan bajo.

## Combining precision and recall – $F_1$ score

	Precision (P)	Recall (R)	$F_1$
Model 1	88.3	79.1	83.4% ←
Model 2	97.0	<u>7.3</u>	13.6%

$$F_1 = \frac{2}{\frac{1}{P} + \frac{1}{R}}$$

- A veces se tiene un modelo con un mejor recuerdo y otro modelo diferente con una mejor precisión. ¿Cómo se comparan dos modelos diferentes?
- Hay una forma común de combinar la precisión y la recuperación utilizando esta fórmula, que se llama puntuación F1. Una de las intuiciones detrás de la puntuación F1 es que quieres que un algoritmo lo haga bien tanto en precisión como en recuperación, y si lo hace peor en precisión o recuperación, eso es bastante malo.

- Para su aplicación, puede tener una ponderación diferente entre la precisión y la recuperación, por lo que F1 no es la única forma de combinar la precisión y la recuperación, es sólo una métrica que se utiliza comúnmente para muchas aplicaciones.

## Multi-class metrics

Classes: Scratch, Dent, Pit mark, Discoloration

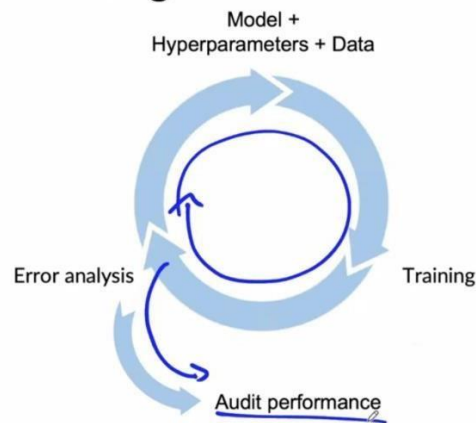
Defect Type	Precision	Recall	$F_1$
Scratch	82.1%	99.2%	89.8%
Dent	92.1%	99.5%	95.7%
Pit mark	85.3%	98.7%	91.5%
Discoloration	72.1%	97%	82.7%

- Hasta ahora, hemos hablado del problema de clasificación binaria con conjuntos de datos sesgados.
- Resulta que también suele ser útil para los problemas de clasificación multiclase.
- Si se trata de detectar defectos en los teléfonos inteligentes, es posible que desee detectar arañazos en ellos, abolladuras, marcas de picaduras o decoloración de la pantalla LCD del teléfono móvil
- Tal vez los cuatro defectos sean en realidad bastante raros como para querer desarrollar un algoritmo que pueda detectarlos todos.
- Una forma de evaluar el rendimiento de su algoritmo en estos cuatro defectos, que pueden ser bastante raros, sería observar la precisión y la recuperación de cada uno de estos cuatro tipos de defectos por separado.
- En el sector de la fabricación, muchas fábricas querrán una gran retirada de productos porque no quieren que un teléfono defectuoso salga a la venta.
- Pero si un algoritmo tiene una precisión ligeramente inferior, no pasa nada, porque a través de un humano que vuelva a examinar el teléfono, es de esperar que se dé cuenta de que el teléfono está realmente bien, por lo que **muchas fábricas harán hincapié en una alta recuperación.**
- Combinando la precisión y la recuperación mediante F1, se obtiene una métrica de evaluación de un solo número para saber lo bien que lo hace el algoritmo en los cuatro tipos de defectos diferentes, y también puede ayudar a establecer una referencia con el rendimiento a nivel humano y también a priorizar en qué trabajar a continuación.
- En lugar de ser preciso con los arañazos, las abolladuras, las marcas de picaduras y las decoloraciones, el uso de la puntuación F1 puede ayudarle a priorizar el tipo de defecto más fructífero en el que intentar trabajar.
- La razón por la que usamos F1 es porque quizás los cuatro defectos son muy raros, por lo que la precisión sería muy alta incluso si el algoritmo se saltara muchos de estos defectos.
- Estas herramientas le ayudarán tanto a evaluar su algoritmo como a priorizar en qué trabajar, tanto en problemas con conjuntos de datos sesgados como en problemas con múltiples clases raras.

## Auditoría de resultados

- Incluso cuando su algoritmo de aprendizaje está funcionando bien en cuanto a precisión o puntuación F1, o alguna métrica apropiada, a menudo merece una última auditoría de rendimiento antes de pasarlo a producción.
- Y esto a veces puede ahorrarle importantes problemas posteriores a la implantación.

## Performance auditing



- Después de haber dado varias vueltas a este ciclo para desarrollar un buen algoritmo de aprendizaje. Vale la pena auditar este rendimiento una última vez

## Auditing framework

Check for accuracy, fairness/bias, and other problems.

1. Brainstorm the ways the system might go wrong.
  - Performance on subsets of data (e.g., ethnicity, gender).
  - How common are certain errors (e.g., FP, FN).
  - Performance on rare classes.
2. Establish metrics to assess performance against these issues on appropriate slices of data.
3. Get business/product owner buy-in.

- He aquí un marco de referencia para comprobar la exactitud, la imparcialidad, la parcialidad y otros posibles problemas de su sistema.
- El primer paso es hacer una lluvia de ideas sobre las diferentes formas en que el sistema podría ir mal.
- Por ejemplo, ¿se comporta el algoritmo suficientemente bien con diferentes subconjuntos de datos, como individuos de una determinada etnia o individuos de diferentes géneros? ¿Comete el algoritmo ciertos errores, como falsos positivos y falsos negativos? ¿Cómo funciona en los casos raros e importantes?
- A continuación, puedes establecer métricas para evaluar el rendimiento de tu algoritmo en función de estos aspectos. Uno de los patrones de diseño más comunes que se ven es que a menudo se evalúa el rendimiento en trozos de los datos.
- Así pues, en lugar de evaluar el rendimiento de todo el conjunto de datos, se pueden eliminar todos los individuos de una determinada etnia, todos los individuos de un determinado sexo o todos los ejemplos en los que hay un defecto de rayado en el teléfono inteligente (también denominados trozos de datos).
- Después de establecer las métricas apropiadas, las herramientas de MLOps pueden ayudar a desencadenar una evaluación automática para cada modelo para auditar este rendimiento. Por ejemplo, Tensorflow tiene un paquete llamado Tensorflow model analysis (TFMA) que calcula métricas detalladas sobre nuevos modelos en diferentes cortes de datos.

- Y como parte de este proceso, también le aconsejaría que obtuviera la aprobación del negocio o del propietario del producto, haciéndole saber que este es el conjunto de problemas más apropiado del que preocuparse, y que hay un conjunto razonable de métricas para evaluar estos posibles problemas.

## Speech recognition example

### 1. Brainstorm the ways the system might go wrong.

- Accuracy on different genders and ethnicities.
- Accuracy on different devices.
- Prevalence of rude mis-transcriptions.

GAN      gun    gang

### 2. Establish metrics to assess performance against these issues on appropriate slices of data.

- Mean accuracy for different genders and major accents.
- Mean accuracy on different devices.
- Check for prevalence of offensive words in the output.

- Si construyes un sistema de reconocimiento de voz, puedes pensar en la forma en que el sistema puede fallar. Una de las cosas que he mirado es la precisión en diferentes géneros y diferentes etnias.
- Un tipo de análisis que también he hecho antes es el de llevar a cabo un análisis de nuestra precisión en función del acento percibido del hablante, porque queremos entender si el rendimiento de los sistemas de voz era una gran función del acento del hablante
- También te puede preocupar la precisión en diferentes dispositivos, ya que cada uno de ellos puede tener un micrófono diferente.
- Otro problema es la prevalencia de errores de transcripción groseros. He aquí un ejemplo de algo que ocurrió en algunos de los cursos de DeepLearning.AI. Uno de nuestros instructores, Laurence Maroney, estaba hablando de GANs, redes generativas adversariales, pero el sistema de transcripción estaba transcribiendo mal GANs porque esto, desafortunadamente, no es una palabra común en el idioma inglés. Así que los subtítulos contenían un montón de referencias a "gun" y "gang", que eran transcripciones erróneas de lo que realmente dijo el instructor, que es GANs.
- Por lo tanto, debemos prestar especial atención a evitar los errores de transcripción que hacen que el sistema de voz piense que alguien ha dicho una palabrota, cuando en realidad no la ha dicho
- A continuación, se pueden establecer métricas para evaluar el rendimiento en relación con estas cuestiones en los trozos de datos adecuados. Por ejemplo, se puede medir la precisión media del sistema de voz para diferentes géneros y para diferentes acentos representados en el conjunto de datos y también comprobar la precisión en diferentes dispositivos y comprobar si hay palabras ofensivas o raíces en la salida.
- Me parece que las formas en que un sistema puede ir mal resultan ser muy dependientes del problema.
- Los distintos sectores y las distintas tareas tendrán normas muy diferentes. Esas normas siguen evolucionando en la IA y en muchos sectores específicos.
- Así que le aconsejo que busque en su sector lo que es aceptable y que se mantenga al día con las normas de imparcialidad y con toda nuestra creciente concienciación sobre cómo hacer que nuestros sistemas sean más justos y menos parciales.
- Sería bueno contar con un equipo o, a veces, incluso con asesores externos que le ayuden a hacer una lluvia de ideas, para reducir el riesgo de que usted o su equipo se vean sorprendidos por algo en lo que no habían pensado.



## Desarrollo de IA centrada en los datos

### Data-centric AI development

#### Model-centric view

Take the data you have, and develop a model that does as well as possible on it.

Hold the data fixed and iteratively improve the code/model.

#### Data-centric view

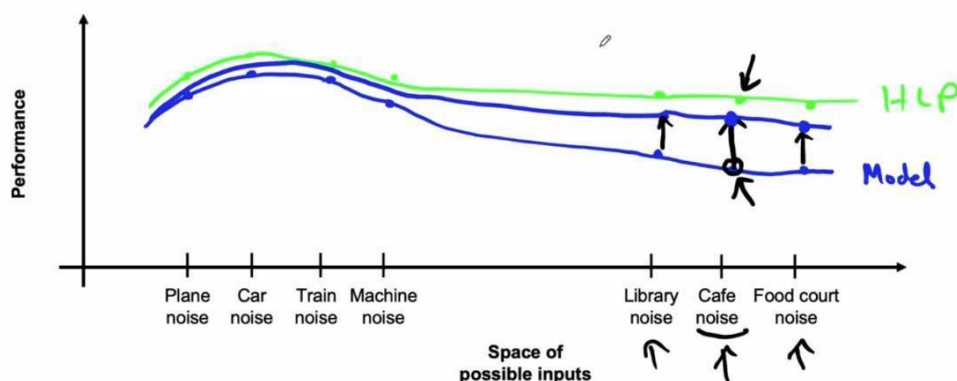
The quality of the data is paramount. Use tools to improve the data quality; this will allow multiple models to do well.

*Hold the code fixed and iteratively improve the data.*

- Con una visión centrada en el modelo de los desarrollos de la IA, usted tomaría los datos que tiene y luego trataría de trabajar muy duro para desarrollar un modelo que lo haga lo mejor posible en los datos
- Esto se debe a que gran parte de la investigación académica sobre la IA fue impulsada por los investigadores, que descargaron un conjunto de datos de referencia y trataron de hacerlo bien en ese punto de referencia.
- La mayor parte de la investigación académica sobre IA se centra en el modelo, porque el conjunto de datos de referencia es una cantidad fija. Desde este punto de vista, el desarrollo centrado en el modelo, mantendría los datos fijos y mejoraría iterativamente el código o el modelo.
- Todavía hay un papel importante que desempeñar para tratar de conseguir mejores modelos, pero esa es una visión diferente de los desarrollos de la IA que creo que es más útil para muchas aplicaciones.
- Se trata de pasar un poco de una visión centrada en el modelo a otra más centrada en los datos.
- Desde este punto de vista, consideramos que la calidad de los datos es primordial, y se pueden utilizar herramientas como el análisis de errores o el aumento de datos para mejorar sistemáticamente la calidad de los datos.
- Para muchas aplicaciones, me parece que si los datos son lo suficientemente buenos, hay varios modelos que sirven perfectamente.
- Desde este punto de vista, se puede mantener fijo el código y mejorar iterativamente los datos. Hay un papel para el desarrollo centrado en el modelo, y hay un papel para el desarrollo centrado en los datos.
- Si ha estado acostumbrado a pensar en el modelo como base de su experiencia con el aprendizaje automático, le insto a que considere la posibilidad de adoptar también una visión centrada en los datos.

## Una imagen útil del aumento de datos

### Speech recognition example

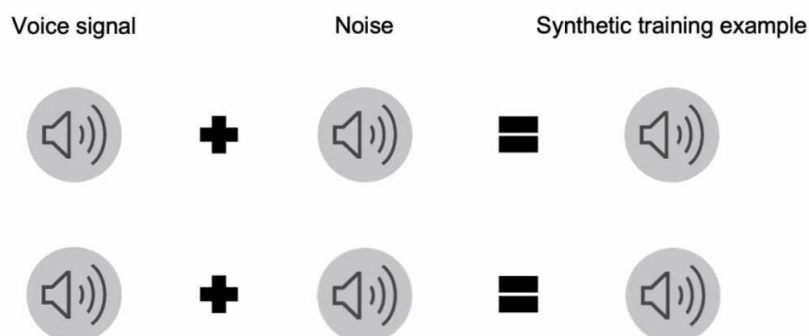


- En este diagrama, el eje vertical representa el rendimiento, digamos la precisión. Y en el eje horizontal hay una especie de eje conceptual de tipo experimento mental, que representa el espacio de las posibles entradas.
- Por ejemplo, tenemos el habla con el ruido de los coches y el ruido de los aviones y el ruido de los trenes, todos ellos bastante similares con el ruido de las máquinas (por ejemplo, lavadoras, aires acondicionados).
- Por otro lado, está el habla con el ruido de la cafetería, el ruido de la biblioteca o el de la sala de comidas, y estos son más similares entre sí que los otros tipos de ruido mecánico.
- Su sistema tendrá diferentes niveles de rendimiento en estos diferentes tipos de entrada.
- Digamos que el rendimiento de los datos con el ruido de la biblioteca, el ruido de la cafetería y el ruido del patio de comidas es peor que el de los ruidos mecánicos
- Así que pienso en todo esto como si fuera una curva, como un trozo de goma/hoja unidimensional que muestra la precisión de tu sistema de habla en función del tipo de entrada que recibe. Esto está representado por una curva azul.
- Un humano tendrá algún otro nivel de rendimiento en estos diferentes tipos de datos (representados por la línea verde), lo que significa que el rendimiento a nivel humano se representa a través de alguna otra curva.
- Por tanto, esta diferencia entre las dos curvas representa una oportunidad de mejora.
- Al realizar el aumento de datos en tipos específicos de datos, imagine que está agarrando un punto de la banda elástica azul y tirando de ella hacia arriba de esta manera.
- Lo que tenderá a hacer es subir esta lámina de goma en la **región adyacente** también. Así que si el rendimiento en el ruido del café sube, probablemente el rendimiento en los puntos cercanos también subirá
- Resulta que para los problemas de datos no estructurados, es poco probable que tirar hacia arriba de un trozo de esta lámina de goma haga que otro trozo de la lámina de goma se sumerja muy por debajo.
- En cambio, si se tira hacia arriba de un punto, los puntos cercanos suben bastante y los lejanos pueden subir un poco, o si se tiene suerte, quizá más que un poco.
- Pero cuando estoy planificando cómo mejorar el rendimiento de mi algoritmo de aprendizaje y hasta dónde espero llegar, y obteniendo más datos en esos lugares para tirar iterativamente con esas piezas o esas partes de la hoja de goma para acercarlas al rendimiento de nivel humano.
- El análisis de los errores le indicará la ubicación de esta nueva brecha más grande, que puede entonces merecer su esfuerzo, para recopilar más datos y, por lo tanto, tratar de arrancar una pieza a la vez.

## Aumento de datos

- El aumento de datos puede ser una forma muy eficaz de obtener más datos, especialmente para los problemas de datos no estructurados, como imágenes, audio o tal vez texto.
- Pero cuando se lleva a cabo el aumento de datos, hay que tomar muchas decisiones. ¿Cuáles son los parámetros? ¿Cómo se diseña la configuración del aumento de datos?

## Data augmentation example




- Por ejemplo, el reconocimiento del habla. Dado un clip de audio, y tomando el ruido de fondo de un café, y sumando estos dos clips de audio (es decir, literalmente, tomar las dos formas de onda y

sumarlas), se puede crear un ejemplo sintético que suena como si alguien estuviera hablando en un café ruidoso

- Al llevar a cabo el aumento de datos, hay que tomar algunas decisiones, como el tipo de ruido de fondo y la intensidad del mismo en relación con el habla.

## Data augmentation

Goal:

Create realistic examples that (i) the algorithm does poorly on, but (ii) humans (or other baseline) do well on 

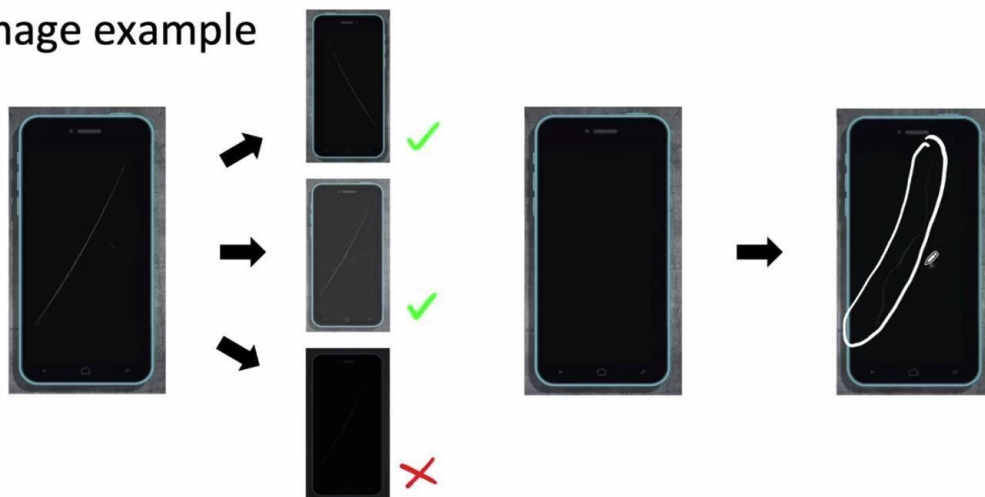
Checklist:

- Does it sound realistic?
  - Is the  $x \rightarrow y$  mapping clear? (e.g., can humans recognize speech?)
  - Is the algorithm currently doing poorly on it?
- 
- El objetivo del aumento de datos es crear ejemplos de los que el algoritmo de aprendizaje pueda aprender.
  - Como marco para hacerlo, te animo a que pienses en cómo puedes crear ejemplos realistas **en los** que el algoritmo lo **haga mal**, porque si el algoritmo ya lo hace bien en esos ejemplos, entonces hay menos de lo que puede aprender.
  - Pero quieres que los ejemplos sigan siendo aquellos en los que un humano o tal vez alguna otra línea de base pueda hacerlo bien, porque de lo contrario, una forma de generar ejemplos sería simplemente crear ejemplos que sean tan ruidosos que nadie pueda escuchar lo que alguien dijo (lo cual no es útil)
  - Quieres ejemplos que sean lo suficientemente difíciles como para desafiar al algoritmo, pero no tan difíciles como para que sea imposible que un humano o un algoritmo lo hagan bien.
  - Ahora bien, una de las formas en que algunas personas hacen el aumento de datos es generar un conjunto de datos aumentados, y luego entrenar el algoritmo de aprendizaje y ver si el algoritmo lo hace mejor en el conjunto dev.

A continuación, juegan con los parámetros de aumento de datos y vuelven a cambiar el algoritmo de aprendizaje, y así sucesivamente. Esto resulta bastante ineficaz porque cada vez que se cambian

- los parámetros de aumento de datos, hay que volver a entrenar el algoritmo.
- En cambio, el uso de estos principios le permite comprobar que los nuevos datos generados mediante el aumento de datos son útiles sin tener que dedicar tal vez horas o días a entrenar un algoritmo de aprendizaje en esos datos para verificar que se producirá la mejora del rendimiento.
- En concreto, esta es una lista de comprobación que podría revisar cuando genere nuevos datos.
- **Uno**, si suena realista. Quieres que tu audio suene realmente como un audio realista del tipo que quieres que tu algoritmo realice.
- **Dos**, ¿está claro el mapeo de X a Y? En otras palabras, ¿los humanos pueden seguir reconociendo lo que se dijo?
- **Tres**, es el algoritmo actualmente haciendo mal en estos nuevos datos. Eso te ayuda a verificar el punto uno.
- Si puedes generar datos que cumplan estos criterios, tendrás más posibilidades de que cuando pongas estos datos en tu conjunto de entrenamiento y vuelvas a entrenar el algoritmo, obtengas mejores resultados

### Image example



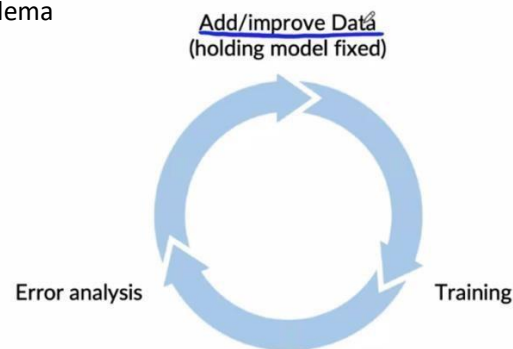
Veamos otro ejemplo, esta vez con imágenes. Supongamos que tienes un conjunto muy pequeño de

- imágenes de smartphones con arañazos. Así es como puedes utilizar el aumento de datos.
- Puedes tomar la imagen y voltearla horizontalmente. El resultado es una imagen bastante realista, y podría ser un ejemplo útil para añadir a su conjunto de entrenamiento.
- O podrías aplicar cambios de contraste para aclarar la imagen aquí, para que el arañazo sea un poco más visible.

- O puedes intentar oscurecer la imagen, pero en este ejemplo del fondo, la imagen es ahora tan oscura que incluso yo como persona no puedo decir realmente si hay un rasguño allí o no.
- Mientras que estos dos ejemplos de arriba pasarían la lista de control que teníamos antes, este ejemplo es demasiado oscuro y fallaría esa lista de control.
- Intentaría elegir el esquema de aumento de datos que genere más ejemplos que se parezcan a los de arriba y pocos de los que se parezcan a los de abajo.
- De hecho, partiendo del principio de que queremos imágenes que parezcan realistas, en las que los humanos puedan hacerlo bien y en las que, con suerte, el algoritmo lo haga mal, también se pueden utilizar técnicas más sofisticadas, como tomar una foto de un teléfono sin arañazos y utilizar Photoshop para dibujar artificialmente un arañazo.
- Esta técnica, utilizando literalmente Photoshop, también puede ser una forma eficaz de generar ejemplos más realistas
- También he utilizado técnicas más avanzadas como GANs para sintetizar arañazos como estos de forma automática, aunque he encontrado que las técnicas son excesivas
- Hay técnicas más sencillas que son mucho más rápidas de implementar y que funcionan muy bien sin la complejidad de construir una GAN para sintetizar arañazos.

## Data iteration loop

es un gran problema



- Es posible que haya oído hablar del término iteración de modelos, que se refiere al entrenamiento iterativo de un modelo utilizando el análisis de errores y luego tratando de decidir cómo mejorar el modelo.
- Si se adopta un enfoque centrado en los datos, resulta útil utilizar un bucle de iteración de datos en el que se toman repetidamente los datos y el modelo, se entrena el algoritmo de aprendizaje, se realiza un análisis de errores y, a medida que se avanza en este bucle, se estudia cómo añadir datos o mejorar la calidad de los mismos.
- Para muchas aplicaciones prácticas, la adopción de este enfoque de bucle de iteración de datos (junto con una búsqueda robusta de hiperparámetros) da lugar a mejoras más rápidas en el rendimiento de su algoritmo de aprendizaje, dependiendo de su problema.

## ¿Añadir datos puede hacer daño?

- Para muchos problemas de aprendizaje automático, los conjuntos de entrenamiento y la distribución de los conjuntos de desarrollo y prueba comienzan siendo razonablemente similares.
- Pero, si estás usando el aumento de datos, estás añadiendo a partes específicas del conjunto de entrenamiento como añadir muchos datos con ruido de café (es decir, obtener datos en áreas en las que el algoritmo no lo está haciendo bien)
- Así que ahora tu conjunto de entrenamiento puede provenir de una distribución muy diferente a la del conjunto de desarrollo y el conjunto de prueba. ¿Va a perjudicar esto el rendimiento de tu álbum de aprendizaje?
- Por lo general, la respuesta es **no** con algunas salvedades cuando se trabaja con problemas de datos no estructurados

## Can adding data hurt performance?

For unstructured data problems, if:

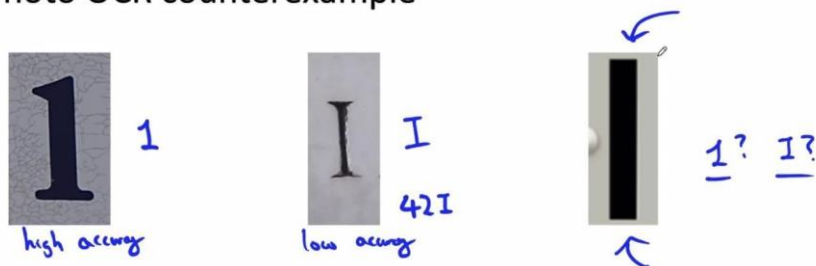
- The model is large (low bias).
- The mapping  $x \rightarrow y$  is clear (e.g., given only the input  $x$ , humans can make accurate predictions).

Then, adding data rarely hurts accuracy.

$P(x)$       20% Cate  
                    ↪ 50%

- Si está trabajando en un problema de datos no estructurados y si su modelo es grande, como una red neuronal que es bastante grande y tiene una gran capacidad y tiene bajo sesgo, entonces no es un gran problema.
- Además, si el mapa de  $x$  a  $y$  es claro (es decir, si sólo se da la entrada  $x$ , los humanos pueden hacer predicciones precisas de  $y$ ), resulta que añadir datos etiquetados con precisión rara vez perjudica la precisión.
- Se trata de una observación importante, ya que añadir datos mediante el aumento de los mismos o recoger más de un tipo de datos puede cambiar la distribución de los datos de entrada.
- Supongamos que al principio de tu problema, el 20% de tus datos tenían ruido de café. Pero al utilizar el aumento, añadiste una gran cantidad de ruido de café para completar el 50%.
- Mientras su modelo sea lo suficientemente grande, no impedirá que haga un buen trabajo con los datos de ruido de los cafés, **así como** con los datos de ruido que no son de los cafés.
- Por el contrario, si su modelo es **pequeño**, cambiar la distribución de los datos de entrada de esta manera puede hacer que gaste demasiados recursos en modelar las configuraciones de ruido de los cafés, y esto podría perjudicar este rendimiento en los datos de ruido que no son cafés.

### Photo OCR counterexample



Adding a lot of new "I"s may skew the dataset and hurt performance

El segundo problema que puede surgir es que el mapeo de  $x$  a  $y$  no sea claro, es decir, que dada  $x$ , la verdadera etiqueta de  $y$  sea muy ambigua. Esto no ocurre mucho en el reconocimiento del habla,

- pero permíteme ilustrarlo con un ejemplo de visión por ordenador.



- Esto es muy raro, así que no es algo que me preocupe por los problemas más prácticos, pero veamos por qué es importante.
- Uno de los sistemas en los que trabajé hace muchos años era sobre las imágenes de Google Street View, con el objetivo de leer los números de las casas para poder localizarlas con mayor precisión en los mapas de Google.
- Así que una de las cosas que hizo el sistema fue tomar fotos de entrada como esta y averiguar los dígitos.
- Así que claramente este es un número uno (imagen de la izquierda) y este es un alfabeto l (imagen del medio).
- No se ven muchas "l" en las imágenes de Street View, ya que los números de las casas rara vez tienen una "l" en el alfabeto.
- Tenemos un algoritmo que tiene una precisión muy alta en el reconocimiento de los números unos, pero una precisión baja en el reconocimiento de las íes.
- Una cosa que podrías hacer es añadir muchos más ejemplos de yoes en tu conjunto de entrenamiento.
- El problema viene (aunque raro) es que algunas imágenes son realmente ambiguas (imagen de la derecha). ¿Es un uno o es una l?
- Y si se añaden muchas "l" nuevas al conjunto de entrenamiento, especialmente ejemplos ambiguos como estos, entonces eso puede sesgar los conjuntos de datos para tener muchas más "l".
- Esto perjudicará el rendimiento porque sabemos que hay muchos más unos que íes en los números de las casas.
- Por lo tanto, sería más seguro adivinar que se trata de un uno y no de un yo.
- Pero si el aumento de datos sesga el conjunto de datos en la dirección de tener muchas más íes en lugar de muchos unos, lo que puede hacer que la red neuronal adivine mal en ejemplos ambiguos como éste.
- Así que este es un raro ejemplo en el que añadir más datos podría perjudicar el rendimiento. En particular, con una imagen como la de la derecha, ni siquiera un ser humano puede decir qué es esto.
- Para que quede claro, el ejemplo que acabamos de ver juntos es un caso bastante raro y es bastante inusual que el aumento de datos o la adición de más datos perjudique el rendimiento de tu algoritmo de aprendizaje, siempre y cuando tu modelo sea lo suficientemente grande.

## Añadir características

- Hasta ahora, nuestro debate se ha centrado en los problemas de datos no estructurados. ¿Qué hay de los problemas de datos estructurados? Resulta que hay un conjunto diferente de técnicas que son útiles para los datos estructurados.
- Para muchos problemas de datos estructurados, la creación de nuevos ejemplos de entrenamiento es difícil, pero hay algo más que se puede hacer, que es tomar los ejemplos de entrenamiento existentes y averiguar si hay características útiles adicionales que se pueden añadir.

### Structured data



#### Restaurant recommendation example

Vegetarians are frequently recommended restaurants with only meat options. ←

Possible features to add?

- Is person vegetarian (based on past orders)?
- Does restaurant have vegetarian options (based on menu)?



- Permítanme utilizar un ejemplo de recomendaciones de restaurantes en el que se quiere recomendar restaurantes a los usuarios
- Una forma de hacerlo sería tener un conjunto de características para cada usuario, y un conjunto de características para cada restaurante. A continuación, se introducen en algunos algoritmos de aprendizaje (por ejemplo, una red neuronal), que luego predice si se trata de una buena recomendación o no,
- En este ejemplo concreto, que es un ejemplo real, el análisis de errores demostró que, por desgracia, el sistema recomendaba con frecuencia a los vegetarianos restaurantes que sólo tenían opciones de carne.
- Así que no fue una buena experiencia para nadie y hubo un fuerte deseo de cambiar esto.
- Ahora bien, no sabía cómo sintetizar nuevos ejemplos de usuarios o nuevos ejemplos de restaurantes porque esta aplicación tenía un conjunto fijo de usuarios y sólo hay un número determinado de restaurantes.
- Así que, en lugar de intentar utilizar el aumento de datos para crear nuevas personas o restaurantes para alimentar el conjunto de entrenamiento, pensé que era **más fructífero ver si había características que añadir** a las entradas de las personas o de los restaurantes.
- En concreto, una característica que puedes considerar añadir es una que indique si esta persona parece ser vegetariana. Y esto no tiene por qué ser una característica de valor binario 0/1. Podría ser una característica como el porcentaje de comida pedida que era vegetariana, o alguna otra medida de la probabilidad de que parezca ser vegetariana.
- Y una característica a añadir en la parte del restaurante sería si tiene opciones vegetarianas
- En el caso de los problemas de datos estructurados, normalmente se tiene un conjunto fijo de usuarios o un conjunto fijo de restaurantes o un conjunto fijo de productos, lo que dificulta el uso del aumento de datos o la recopilación de nuevos datos de nuevos usuarios que aún no se tienen sobre restaurantes que pueden o no existir.
- En cambio, añadir características, puede ser una forma más fructífera de mejorar el rendimiento del algoritmo para solucionar problemas como éste, identificado a través del análisis de errores.
- Características adicionales como estas, pueden ser codificadas a mano o podrían a su vez ser generadas por algún algoritmo de aprendizaje, como tener un algoritmo de aprendizaje que intente leer el menú y clasificar las comidas como vegetarianas o no

## Other food delivery examples

- Only tea/coffee
- Only pizza

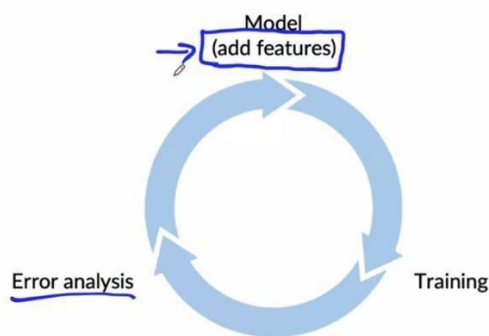
What are the added features that can help make a decision?

Product recommendation:



- En cuanto a los ejemplos de entrega de comida, descubrimos que había algunos usuarios que sólo pedían té y café y otros que sólo pedían pizza.
- Así que si el equipo de producto quiere mejorar la experiencia de estos usuarios, un equipo de aprendizaje automático podría preguntarse cuáles son las características adicionales que podemos añadir (es decir, enriquecer las características) para detectar quiénes son las personas que sólo piden té o café o quiénes son las personas que sólo piden la pizza.
- En los últimos años, ha habido una tendencia en las recomendaciones de productos de **pasar de enfoques de filtrado colaborativo a enfoques de filtrado basados en el contenido**.
- Los enfoques de filtrado colaborativo son, a grandes rasgos, un enfoque que mira al usuario, trata de averiguar quién es similar a ese usuario y luego le recomienda cosas que a la gente como usted también le han gustado.
- Por el contrario, un enfoque de filtrado basado en el contenido tenderá a mirarle a usted como persona y a fijarse en la descripción del restaurante o a mirar el menú (u otra información) de los restaurantes, y entonces verá si ese restaurante es una buena opción para usted o no.
- La ventaja del filtrado basado en el contenido es que incluso si hay un nuevo restaurante o un nuevo producto que no le ha gustado a casi nadie, al mirar realmente la descripción del restaurante en lugar de sólo mirar a quién le gustan los restaurantes, todavía puede hacer rápidamente buenas recomendaciones.
- A veces también se le llama el **problema del arranque en frío**, es decir, ¿cómo recomendar un producto nuevo que casi nadie ha comprado ni le ha gustado hasta ahora?
- Y una de las formas de hacerlo es asegurarse de captar buenas características para las cosas que se quieran recomendar.
- A diferencia del filtrado colaborativo, que requiere que un grupo de personas vean el producto y decidan si les gusta o no antes de poder decidir si se debe recomendar el mismo producto a un nuevo usuario.

## Data iteration



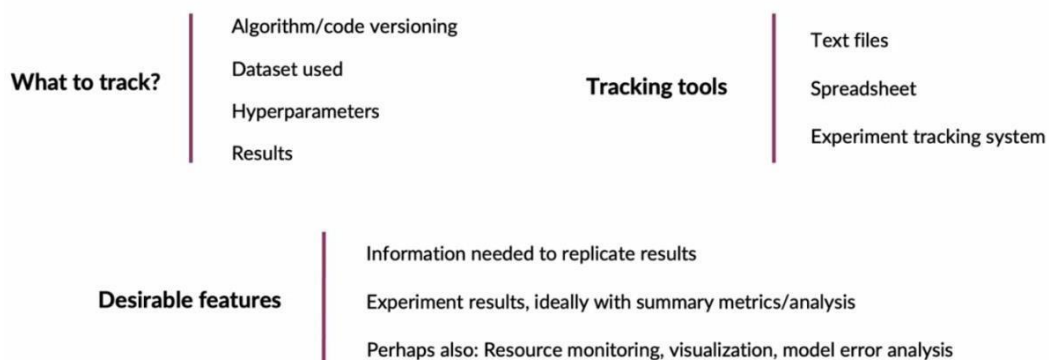
- Error analysis can be harder if there is no good baseline (such as HLP) to compare to.
- Error analysis, user feedback and benchmarking to competitors can all provide inspiration for features to add.

- La iteración de datos para los problemas de datos estructurados puede tener este aspecto.

- Se parte de un modelo, se entrena el modelo y luego se realiza un análisis de errores.
- El análisis de errores puede ser más difícil en los problemas de datos estructurados si **no hay una buena línea de base, como el** rendimiento a nivel humano, para comparar, y el rendimiento a nivel humano es difícil para los datos estructurados porque es realmente difícil que la gente recomiende buenos restaurantes, incluso entre sí.
- Pero descubrí que el análisis de errores, las opiniones de los usuarios y la comparación con los competidores pueden descubrir ideas para mejorar
- A través de estos métodos, si puedes identificar un cierto tipo de etiqueta asociada a tus datos que quieras mejorar, entonces puedes volver a seleccionar algunas características para añadir. Los ejemplos son como las características para averiguar quién es vegetariano y qué restaurantes tienen buenas opciones vegetarianas
- Y como la aplicación puede tener sólo una lista finita de usuarios y restaurantes, los usuarios y restaurantes que tiene pueden ser todos los datos que tiene, añadir características a los ejemplos es un enfoque más fructífero que tratar de llegar a nuevos usuarios o nuevos restaurantes.
- Sé que hace muchos años, antes del auge del aprendizaje profundo, parte de la esperanza del aprendizaje profundo era que ya no había que diseñar a mano las características. Creo que eso se ha hecho realidad en su mayor parte para los problemas de datos no estructurados.
- Cuanto mayor sea el conjunto de datos, más probable será que un algoritmo de aprendizaje profundo puro pueda funcionar.
- Pero incluso con el aumento del aprendizaje profundo, si el tamaño de tu conjunto de datos no es masivo, el diseño de características, especialmente para los datos estructurados, puede seguir siendo un motor muy importante para mejorar el rendimiento.

## Seguimiento del experimento

### Experiment tracking



Mientras trabajas en la mejora iterativa de tu algoritmo, una cosa que te ayudará a ser un poco más eficiente es asegurarte de que tienes un sólido seguimiento de los experimentos.

- Cuando se realizan docenas o cientos, o incluso más, de experimentos, es fácil olvidar los que ya se han realizado.
- Disponer de un sistema de seguimiento de sus experimentos puede ayudarle a ser más eficiente en la toma de decisiones sobre los datos o el modelo o los hiperparámetros para mejorar sistemáticamente el rendimiento de su algoritmo.

- Cuando se hace un seguimiento de los experimentos realizados, es decir, de los modelos que se han entrenado, hay algunas cosas que le insto a seguir.
- Una, es llevar un registro de qué algoritmo estás utilizando y qué versión de código. Mantener un registro de esto hará que sea mucho más fácil para usted volver y replicar el experimento
- En segundo lugar, hay que tener en cuenta el conjunto de datos que se utiliza.
- Tercero, los hiperparámetros y cuarto, guardar los resultados en algún lugar. Esto debería incluir al menos las métricas de alto nivel como la precisión o la puntuación F1 o las métricas relevantes, pero si es posible, sería útil guardar simplemente una copia del modelo entrenado.
- ¿Cómo puede hacer un seguimiento de estas cosas? He aquí algunas herramientas de seguimiento que puede considerar.
- Muchas personas, y a veces incluso equipos, empiezan con **archivos de texto**.
- Cuando hago un experimento por mi cuenta, puede que use un archivo de texto para hacer una nota con unas pocas líneas de texto por experimento para anotar lo que estaba haciendo.
- Esto no se adapta bien, pero puede estar bien para pequeños experimentos.
- Muchos equipos pasan de los archivos de texto a las hojas de cálculo, especialmente a las hojas de cálculo compartidas.
- En realidad, las hojas de cálculo se amplían bastante más, especialmente las hojas de cálculo compartidas que pueden consultar varios miembros de un equipo.
- Algunos equipos también considerarán la posibilidad de migrar a un sistema de seguimiento de experimentos más formal. El espacio de los sistemas de seguimiento de experimentos sigue evolucionando rápidamente, al igual que el creciente conjunto de herramientas existentes. Pero algunos ejemplos son Weight&Biases, Comet, MLflow y SageMaker Studio.
- LandingAI también tiene su propia herramienta de seguimiento de experimentos centrada en la visión por ordenador y las aplicaciones de fabricación.
- Cuando intento utilizar una herramienta de seguimiento, ya sea un archivo de texto o una hoja de cálculo o algún sistema más amplio, estas son algunas de las cosas en las que me fijo.
- En primer lugar, ¿me da toda la información necesaria para replicar los resultados?
- En términos de replicabilidad, una cosa que hay que tener en cuenta es si el algoritmo de aprendizaje extrae datos de Internet. Dado que los datos de Internet pueden cambiar, puede disminuir la replicabilidad a menos que se tenga cuidado en cómo se implementa el sistema.
- En segundo lugar, herramientas que le ayuden a comprender rápidamente los resultados experimentales de un entrenamiento específico, idealmente con métricas de resumen útiles y tal vez incluso un análisis en profundidad,
- Por último, algunas otras características a tener en cuenta, la monitorización de recursos, ¿cuántos recursos de memoria de la CPU/GPU utiliza? ¿O herramientas que le ayuden a visualizar el modelo entrenado o incluso herramientas que le ayuden con un análisis de errores más profundo?
- Sin embargo, en lugar de preocuparse demasiado por el marco de seguimiento de experimentos que hay que utilizar, lo que espero que te lleves de este vídeo es que tienes que tener algún sistema, aunque sea un archivo de texto o una hoja de cálculo, para hacer un seguimiento de tus experimentos e incluir toda la información que sea conveniente.

## From Big Data to Good Data



Try to ensure consistently high-quality data in all phases of the ML project lifecycle.

Good data:

- Covers important cases (good coverage of inputs  $x$ ) ←
  - Is defined consistently (definition of labels  $y$  is unambiguous)
  - Has timely feedback from production data (distribution covers data drift and concept drift)
  - Is sized appropriately
- 
- Gran parte de la IA moderna ha crecido en las grandes empresas de Internet de consumo, con quizás mil millones de usuarios, y estas empresas tienen muchos datos sobre sus usuarios.
  - Si tienes grandes datos como esos, por supuesto, el big data podría ayudar al rendimiento de tu sala tremendamente.
  - Pero para muchas otras industrias, simplemente no hay mil millones de puntos de datos.
  - Y creo que puede ser aún más importante que esas aplicaciones se centren no sólo en los grandes datos, sino en los buenos datos.
  - Descubrí que si se puede garantizar la alta calidad de los datos en todas las fases del ciclo de vida del proyecto de aprendizaje automático, es la clave para asegurarse de tener un despliegue de aprendizaje automático fiable y de alto rendimiento.
  - Los buenos datos cubren los casos importantes, por lo que debes tener una buena cobertura de las diferentes entradas  $x$ . Y si descubres que no tienes suficientes datos, el aumento de datos puede ayudarte a conseguir más datos para obtener entradas  $x$  más diversas y darte esa cobertura.
  - Los buenos datos también se definen de forma coherente con la definición de las etiquetas y no son ambiguos.
  - Los buenos datos también tienen una retroalimentación oportuna de los datos de producción. De hecho, hablamos de esto la semana pasada cuando cubrimos la sección de despliegue en términos de tener sistemas de monitoreo para rastrear la deriva del concepto y la deriva de los datos.
  - Y, por último, se necesita un conjunto de datos de tamaño razonable.

## Lecturas

- [Establecer una línea de base](#)
- [Análisis de errores](#)
- [Seguimiento del experimento](#)
- Brundage, M., Avin, S., Wang, J., Belfield, H., Krueger, G., Hadfield, G., ... Anderljung, M. (s.f.). Toward trustworthy AI development: Mechanisms for supporting verifiable claims\*. Recuperado el 7 de mayo de 2021 <http://arxiv.org/abs/2004.07213v2>
- Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., & Sutskever, I. (2019). Doble descenso profundo: Donde modelos más grandes y más datos hacen daño. Recuperado de <http://arxiv.org/abs/1912.02292>