Curso 2 - Ciclo de vida de los datos de aprendizaje automático en la producción

En el segundo curso de Ingeniería de Aprendizaje Automático para la Especialización de Producción, construirá pipelines de datos mediante la recopilación, limpieza y validación de conjuntos de datos y la evaluación de la calidad de los datos; implementará la ingeniería de características, la transformación y la selección con TensorFlow Extended y obtendrá el mayor poder predictivo de sus datos; y establecerá el ciclo de vida de los datos aprovechando las herramientas de metadatos de linaje y procedencia de los datos y seguirá la evolución de los datos con esquemas de datos empresariales.

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 las funciones modernas de desarrollo de software e ingeniería para ayudarte a desarrollar habilidades listas para la producción.

Semana 1: Introducción al MLEP

Contenido

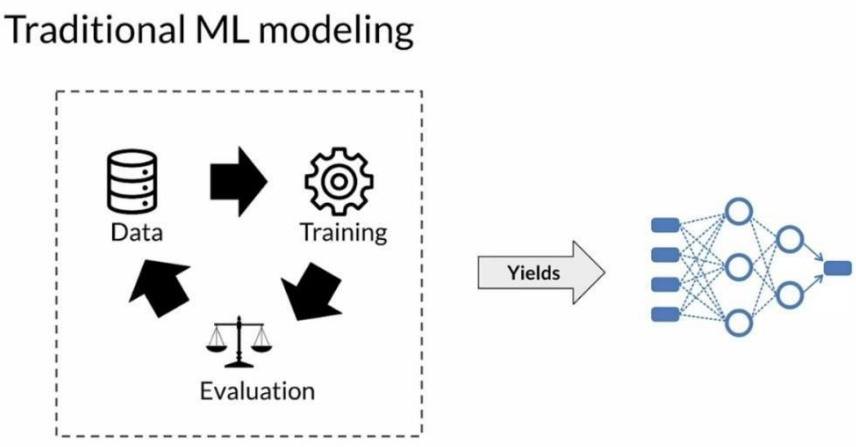
Introducción

The importance of data

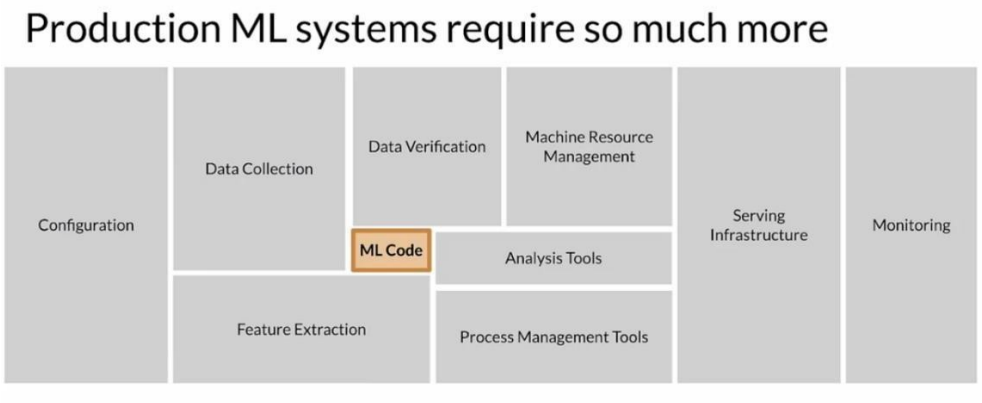
"Data is the hardest part of ML and the most important piece to get right...
Broken data is the most common cause of problems in production ML systems"
- Scaling Machine Learning at Uber with Michelangelo - Uber

"No other activity in the machine learning life cycle has a higher return on investment than improving the data a model has access to."
- Feast: Bridging ML Models and Data - Gojek

- En los entornos de producción, se descubren algunas cosas interesantes sobre la importancia de los datos.
- A continuación se presentan dos citas de profesionales del ML que participan en empresas en las que los datos y el ML son fundamentales.
- En primer lugar, desde Uber, los datos son la parte más difícil de ML y la pieza más importante para hacerlo bien. Los datos rotos son la causa más común de problemas en los sistemas de ML de producción
- Según Gojek, ninguna otra actividad del ciclo de vida del aprendizaje automático tiene un mayor retorno de la inversión que la mejora de los datos a los que tiene acceso un modelo.
- La verdad es que si vas a cualquier equipo de producción de ML y les preguntas sobre la importancia de los datos, obtendrás respuestas similares.
- Por eso hablamos de los datos, porque son increíblemente importantes para el éxito y los problemas de los datos en los entornos de producción son muy diferentes de los entornos académicos o de investigación con los que podrías estar familiarizado.



- En un entorno académico o de investigación, la modelización es realmente bastante sencilla. Bueno, tal vez no sea sencillo, pero sí menos complicado.
- Por lo general, se dispone de un conjunto de datos, a menudo estándar, que se suministran ya limpios y etiquetados, que se van a utilizar para entrenar el modelo y evaluar los resultados.
- El resultado final es un modelo que hace buenas predicciones.
- Así que probablemente pasará por unas cuantas iteraciones para optimizar completamente el modelo. Una vez que estés satisfecho con los resultados, normalmente habrás terminado.



- El ML de producción requiere mucho más que un modelo.
- Descubrimos que el modelo suele representar alrededor del 5% del código necesario para poner en producción una aplicación ML.
- Echa un vistazo a todas las demás cajas que hay en este diagrama y te harás una idea de lo que vamos a hablar.
- Fundamentalmente, no estamos hablando sólo de aprendizaje automático y modelado. Hablamos de aplicaciones ML de producción y de lo que supone crearlas, desplegarlas, mantenerlas y mejorarlas para poder ponerlas a disposición de sus usuarios y de su negocio

ML modeling vs production ML

	Academic/Research ML	Production ML
Data	Static	Dynamic - Shifting
Priority for design	Highest overall accuracy	Fast inference, good interpretability
Model training	Optimal tuning and training	Continuously assess and retrain
Fairness	Very important	Crucial
Challenge	High accuracy algorithm	Entire system

- Comparemos algunas de las diferencias entre el modelado ML en un entorno de investigación/académico y el ML de producción real.
- Para empezar, en un entorno académico o de investigación, se suele utilizar un conjunto de datos estático, mientras que para el ML de producción se utilizan datos del mundo real, que son dinámicos y suelen cambiar.
- La prioridad de diseño para el ML académico o de investigación suele ser la mayor precisión en todo el conjunto de entrenamiento, pero la prioridad de diseño para el ML de producción es la inferencia rápida y la buena interpretabilidad (y, por supuesto, la precisión y el coste).
- El entrenamiento del modelo para el ML de investigación se basa en un único resultado óptimo, centrándose en el ajuste y el entrenamiento necesarios para conseguirlo. En cambio, el ML de producción requiere un seguimiento, una evaluación y un reentrenamiento continuos.
- La interpretabilidad y la imparcialidad son importantes para cualquier modelo de ML, pero son absolutamente cruciales para el ML de producción.
- Por último, mientras que el principal reto del ML académico y de investigación es la puesta a punto de un modelo de alta precisión, el reto del ML de producción es ese más todo lo demás, es decir, todo el sistema.

Production machine learning

Machine learning development + Modern software development

- Sería justo decir que se puede considerar que el aprendizaje automático de producción es tanto el aprendizaje automático en sí mismo como el conjunto de conocimientos y habilidades necesarios para el desarrollo de software moderno.
- Realmente se requiere experiencia en ambas áreas para tener éxito, porque no se está produciendo un único resultado, sino que se está desarrollando un producto o servicio que a menudo es una parte crítica de la oferta.

Managing the entire life cycle of data

- Labeling
- Feature space coverage
- Minimal dimensionality
- Maximum predictive data
- Fairness
- Rare conditions

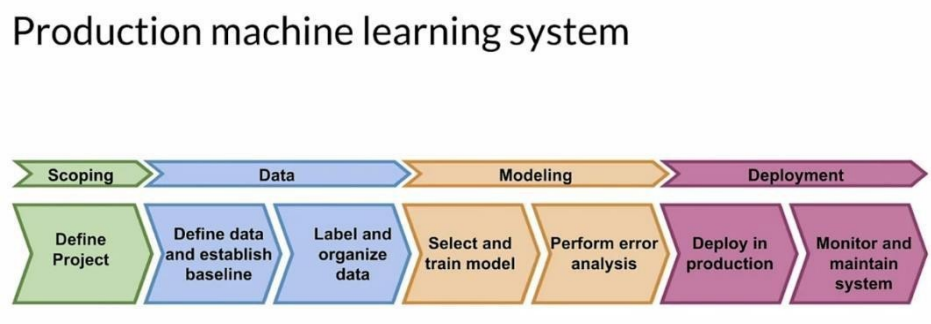
- El propio desarrollo del ML se centra en cuestiones específicas relacionadas con los datos y la calidad de las predicciones.
- Por ejemplo, suponiendo que esté realizando un aprendizaje supervisado, debe asegurarse de que sus etiquetas sean precisas y de que su conjunto de datos de entrenamiento tenga ejemplos que cubran el mismo espacio de características que la solicitud que recibirá su modelo.
- También quiere reducir la dimensionalidad de su vector de características para optimizar el rendimiento de su sistema, al tiempo que conserva o mejora la información predictiva de sus datos.
- A lo largo de todo esto, hay que tener en cuenta y medir la equidad de los datos y el modelo, especialmente para las condiciones raras, por ejemplo, en dominios como la atención sanitaria, donde las condiciones raras pero importantes pueden ser absolutamente críticas para el éxito.

Modern software development

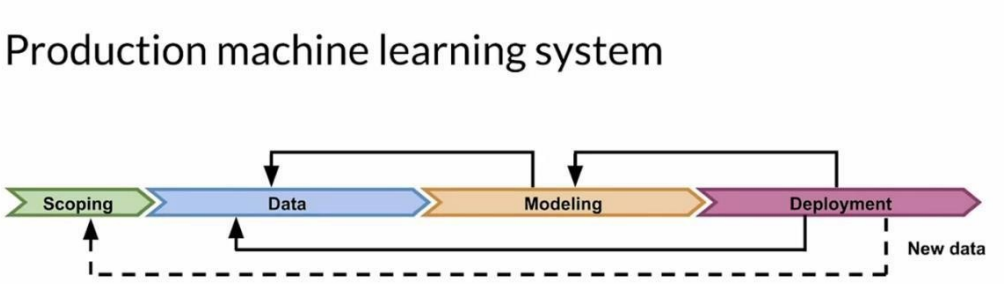
Accounts for:

- Scalability
- Extensibility
- Configuration
- Consistency & reproducibility
- Safety & security
- Modularity
- Testability
- Monitoring
- Best practices

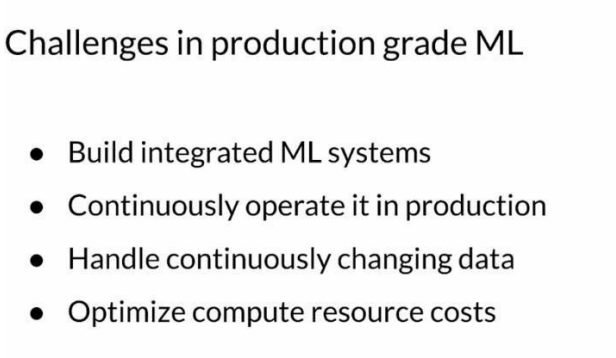
- Además de todo eso, estás poniendo en producción una pieza de software que requiere un diseño de sistema, y eso incluye todas las cosas que se requieren para cualquier despliegue de software de producción.
- Por supuesto, este despliegue tiene que estar centrado en ML y en su aplicación.
- ¿Es su sistema escalable? ¿Se puede ampliar y reducir? ¿Puede ampliarlo de forma limpia para añadir nuevas cosas cuando lo necesite? ¿Tiene una configuración clara y bien definida? ¿Es coherente? ¿Puede reproducir los resultados de forma fiable? ¿Está reforzado contra los ataques? ¿Es el diseño modular y sigue los principios modernos de desarrollo de software?
- ¿Puede probar unidades individuales? ¿Puede realizar pruebas de extremo a extremo? ¿Puede supervisar continuamente la salud y el rendimiento de su sistema y ser alertado cuando hay problemas? ¿Ha adoptado las mejores prácticas del sector?



- El uso de un modelo en aplicaciones del mundo real requiere mucho más que la comprensión de los algoritmos de aprendizaje automático.
- El primer paso es la determinación del alcance, que se centra en definir las necesidades y los objetivos del proyecto y los recursos necesarios para alcanzarlos.
- A continuación, se empieza a trabajar en los datos, lo que incluye definir las características que se van a utilizar, así como organizar y etiquetar los datos. En ocasiones, esto puede incluir la medición del rendimiento a nivel humano para establecer una línea de base para la comparación.
- A continuación, diseñe y entrene un modelo. En esta fase, el análisis de errores le ayudará a perfeccionar su modelo para adaptarlo a las necesidades de su proyecto. Después de entrenar su modelo, lo despliega para que pueda ser utilizado para servir a las solicitudes de predicción.
- Puede desplegar su modelo en dispositivos móviles, en una nube o en dispositivos IoT, o incluso en un navegador web.
- Con el tiempo, los datos del mundo real cambian continuamente, lo que puede provocar una degradación del rendimiento de su modelo.
- Hay que supervisar continuamente el rendimiento del modelo y, si se detecta un descenso en el rendimiento, hay que volver a reentrenar y ajustar el modelo, o revisar los datos.

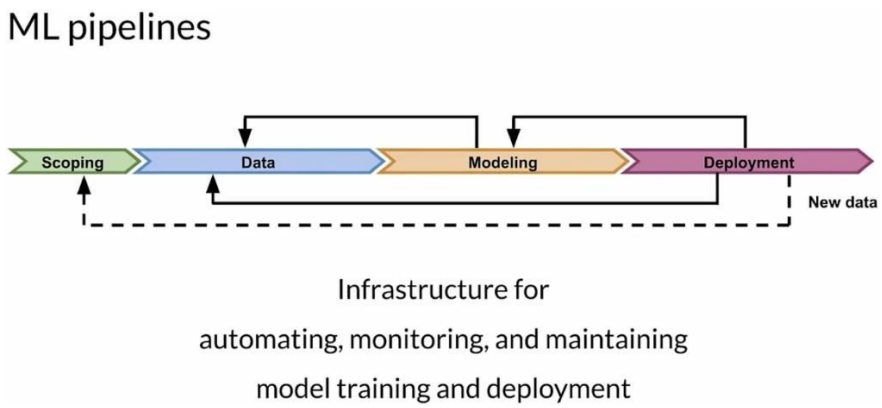


- Durante el despliegue, los nuevos datos pueden afectar al diseño de su proyecto, ya sea positiva o negativamente, y puede ser necesario hacer frente a los riesgos.
- En última instancia, todos estos pasos crean su sistema de producción de ML, que debe ejecutarse de forma automática de manera que se supervise continuamente el rendimiento de su modelo, la ingesta de nuevos datos y el reentrenamiento, según sea necesario, y luego la redistribución para mantener o mejorar su rendimiento.

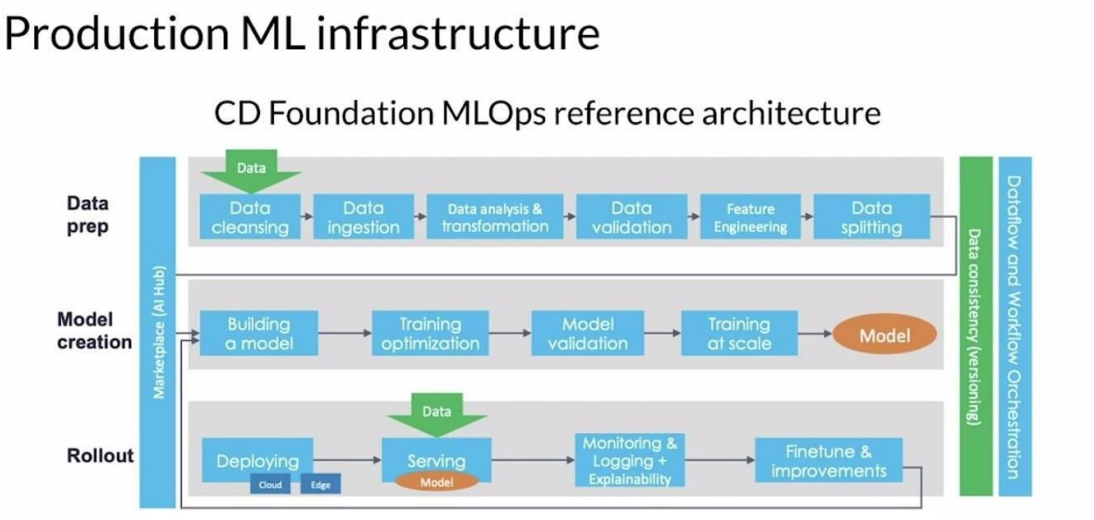


- Los retos al hacer ML de producción son muy diferentes a los del ML académico o de investigación, o en cierto sentido son los mismos pero incluyen mucho más.
- Vas a construir un sistema integrado centrado específicamente en los casos de uso de ML.
- Hay que pensar en operarlo continuamente en producción, y para los casos de uso en línea, eso significa que tiene que estar disponible 24/7.
- Hay que pensar y poner en marcha sistemas para manejar un mundo y unos datos cambiantes y, por supuesto, como cualquier sistema de producción, hay que intentar hacer todo esto con el mínimo coste y produciendo el máximo rendimiento.
- Puede parecer desalentador, pero la buena noticia es que existen herramientas y metodologías bien establecidas para hacerlo.

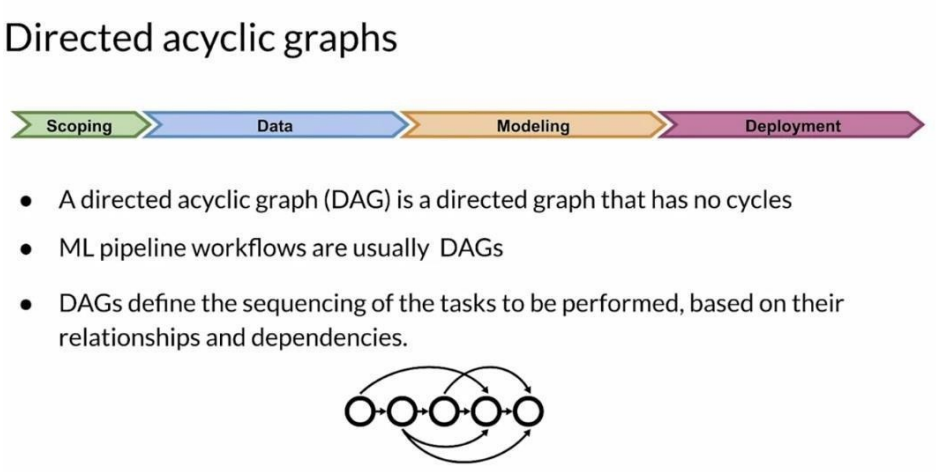
Pipelines de ML



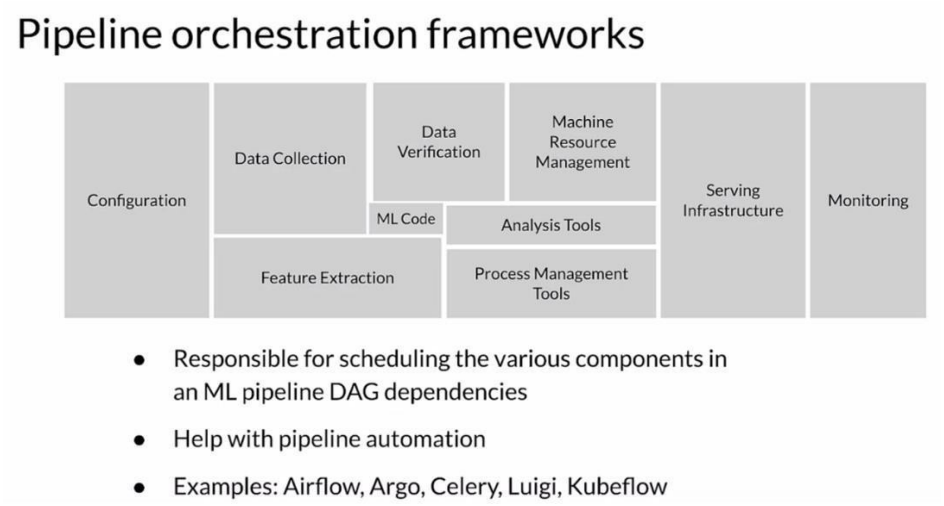
- En esta lección, comenzaremos a introducir los Pipelines ML y el concepto de MLOps.
- También veremos cómo los orquestadores de tuberías secuencian y programan las tareas de ML para implementar todo el proceso de entrenamiento de ML.
- A continuación, veremos el ejemplo de TensorFlow Extended o TFX, que es un marco ampliamente adoptado para crear Pipelines ML.
- ¿A qué nos referimos con la frase ML Pipeline? Bueno, recuerde el flujo de trabajo iterativo de ML del que hablamos anteriormente. ML **Pipeline es una arquitectura de software** para implementar exactamente eso.
- Automatizar, supervisar y mantener este flujo de trabajo de ML desde los datos hasta un modelo entrenado.
- Los pipelines de ML son un componente clave de las arquitecturas de MLOps.
- Los pipelines de ML ofrecen soporte para la automatización, la supervisión y el mantenimiento de un modelo a medida que se sigue entrenando a lo largo de su vida.



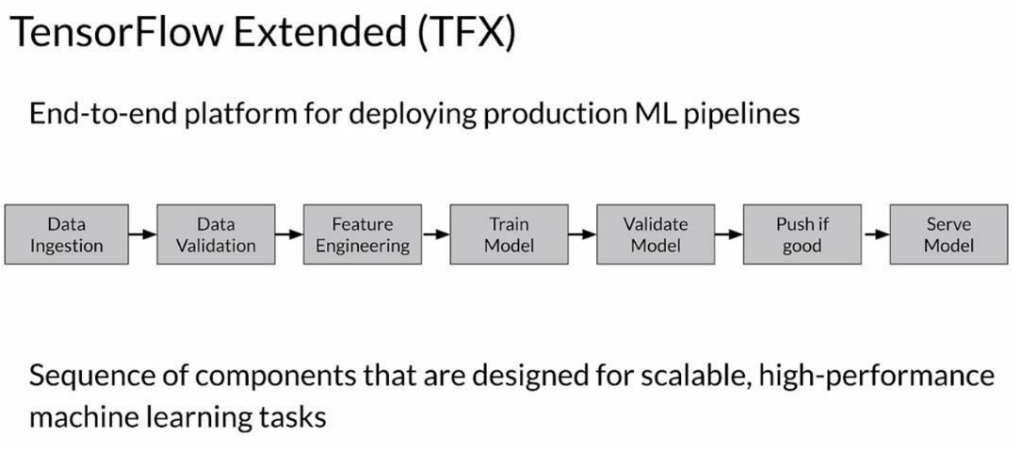
- Esta diapositiva muestra una versión de lo que es una tubería de LD, que fue elaborada por un grupo de la industria, la Fundación CD.
- Hay algunas diferencias entre las distintas arquitecturas de tuberías, pero en general se parecen a esto.
- Verás que básicamente reflejan el proceso de desarrollo de ML. Comenzando con la ingestión de datos y terminando con un modelo entrenado. Eso es por diseño, ya que necesitan encapsular y formalizar ese proceso.



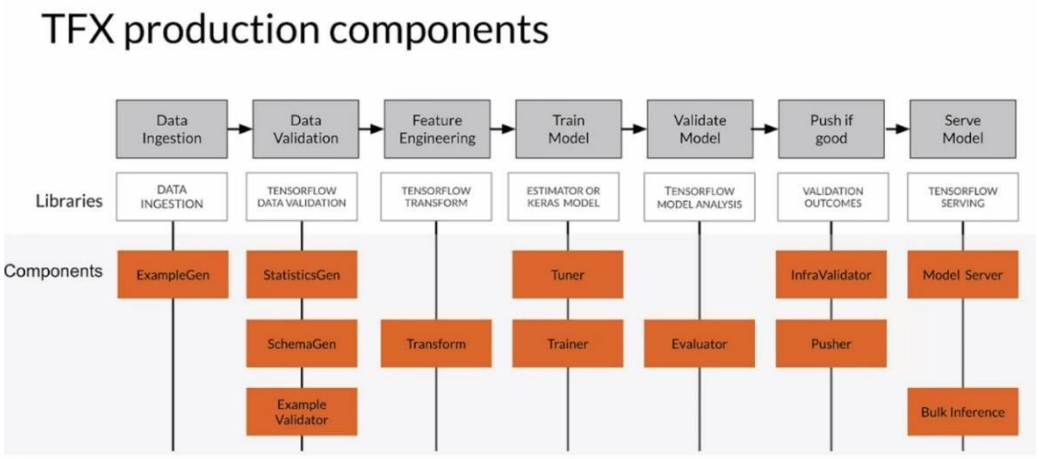
- Los pipelines ML son casi siempre grafos cíclicos dirigidos o DAGs, aunque en algunos casos avanzados pueden incluir ciclos.
- Un DAG es una colección de todas las tareas que quieres ejecutar secuenciadas de forma que reflejen sus relaciones y dependencias.
- Observa que en este grafo las aristas son dirigidas y no hay ciclos. Esto hace que este grafo sea un DAG



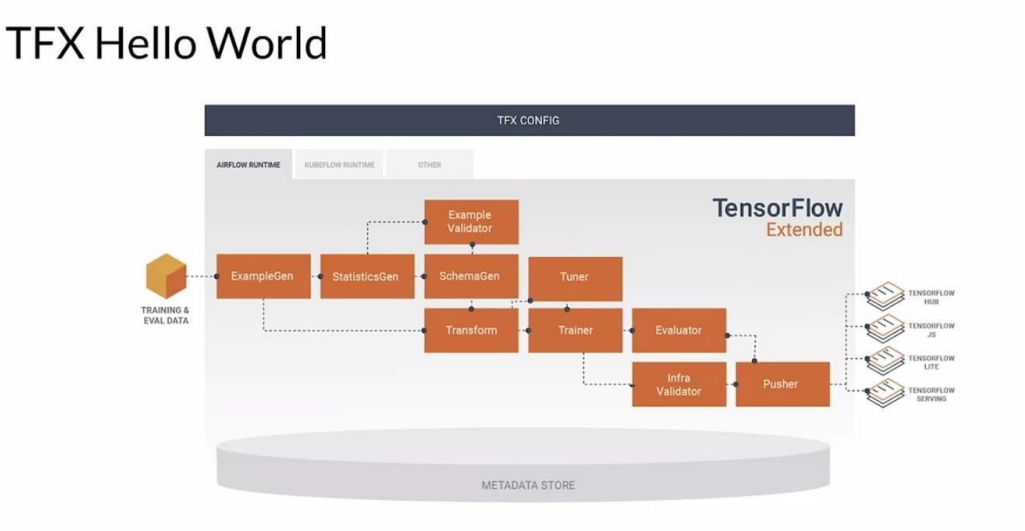
- Los orquestadores se encargan de programar los distintos componentes de un ML Pipeline basándose en las dependencias definidas por un DAG.
- Los orquestadores ayudan a la automatización de las canalizaciones.
- Algunos ejemplos son Argo, Airflow, Celery, Luigi y Kubeflow.



- TFX es una plataforma de aprendizaje automático de código abierto para el despliegue de cadenas de producción de ML, y es lo que usamos en Google.
- Un **TFX Pipeline** es una secuencia de componentes escalables que pueden manejar grandes volúmenes de datos.
- Empezando por la izquierda, ingerimos los datos y luego pasamos a la validación de datos y hacemos algo de ingeniería de características.
- Entonces entrenamos un modelo y lo validamos. Si es mejor que lo que tenemos en producción, lo pasamos a producción.
- Por último, servimos predicciones. La secuencia de componentes está diseñada para tareas de Machine Learning escalables de alto rendimiento.



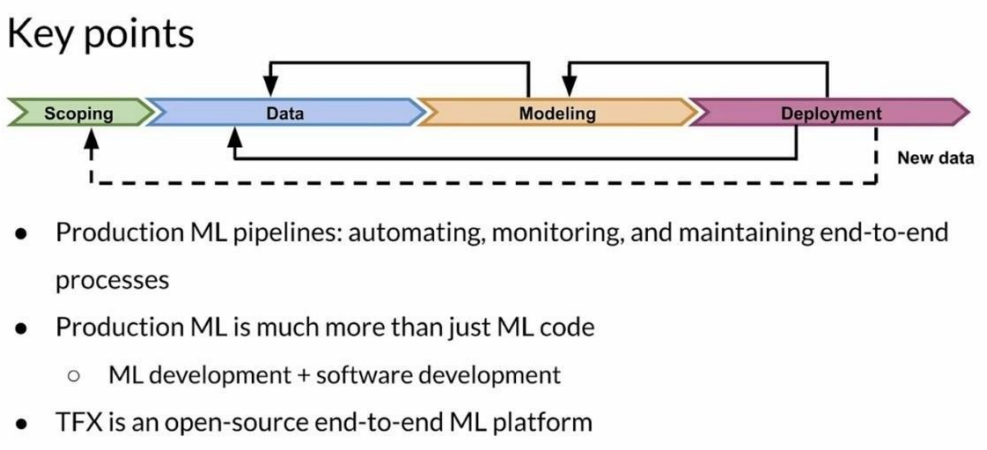
- En este curso, usted usará TFX para implementar Pipelines ML reales como lo haría para Sistemas de Producción.
- Los componentes de TFX en producción están construidos sobre bibliotecas de código abierto, como Tensorflow Data Validation, Tensorflow Transform y Tensorflow Model Analysis.
- Los componentes de color naranja aprovechan esas Bibliotecas y forman su DAG a medida que secuencian estos componentes y establecen la dependencia entre ellos, lo que luego forma su Pipeline ML.



- Esto es lo que llamamos el Hola Mundo de TFX. Comenzamos a la izquierda con nuestros datos y vamos a ingerir nuestros datos con un componente TFX llamado ExampleGen. Todas las cajas en naranja son componentes TFX.
- De hecho, estos son componentes que vienen con el TFX cuando sólo se hace una instalación PIP.
- A continuación, generamos estadísticas para nuestros datos. Queremos saber los rangos de nuestras características, si son características numéricas, si son características categóricas, queremos saber cuáles son las categorías válidas, etc.
- Example Validator se utiliza para buscar problemas en nuestros datos. SchemaGen se utiliza para generar un esquema para nuestros datos a través de nuestro vector de características. Transform hará la ingeniería de características.
- Tuner y Trainer se utilizan para entrenar un modelo y afinar los hiperparámetros de ese modelo.
- El evaluador se utiliza para hacer un análisis profundo del rendimiento de nuestro modelo.
- Infra Validator se utiliza para asegurarse de que realmente podemos ejecutar predicciones utilizando nuestro modelo en la infraestructura que tenemos. Por ejemplo, ¿tenemos suficiente memoria?
- Si todo eso se supera y el modelo funciona realmente mejor que lo que podríamos tener ya en producción, entonces Pusher empuja el modelo a Producción. ¿Qué significa esto? Bueno, podríamos

estar empujando a un repositorio como Tensorflow HUB y luego usar nuestro modelo más tarde para tal vez el aprendizaje de transferencia.

- Podríamos empujar a TensorFlow JS, si vamos a utilizar nuestro modelo en un navegador web o una aplicación Node.js.
- Podríamos empujar a TensorFlow Lite y utilizar nuestro modelo en una aplicación móvil o en un dispositivo IOT.
- También podríamos empujar a TensorFlow Serving y UserModel en un servidor o tal vez en un cluster de servidores.



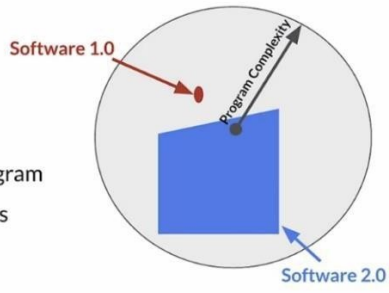
- Los puntos clave aquí, en primer lugar, es que las tuberías de producción de ML son más que el código de ML.
- Son el desarrollo de ML combinado con el desarrollo de software, y son un proceso formalizado para ejecutar esa secuencia de tareas de extremo a extremo, de forma mantenible y escalable.

La importancia de los datos

- En primer lugar, permítanme contarles una historia sobre una aplicación en la que participé.
- Nos pidieron que creáramos un modelo para predecir el tiempo que se tardaría en pasar por el control de seguridad de un aeropuerto en diferentes días y a diferentes horas, con colas de diferente longitud, etc.
- Necesitamos datos, así que tuvimos que medir el tiempo que tardaba la gente en pasar los controles de seguridad.
- Teníamos a una persona al principio de la cola para entrar en un control de seguridad y registraban la hora a la que alguien entraba.
- Y luego teníamos a otra persona de pie en el otro extremo, la salida del puesto de control, y en realidad estaban lo suficientemente lejos el uno del otro. Ni siquiera podían verse el uno al otro, y registraban la hora de salida de cada persona.
- Y así, poco a poco, íbamos construyendo un conjunto de datos etiquetados que nos daban la cantidad de tiempo que la gente tardaba en pasar el control de seguridad de un aeropuerto. Bueno, como puedes imaginar, fue increíblemente doloroso y costoso...
- Así que cuando hablemos de recopilar datos en el mundo real, es de esperar que no te enfrentes a una situación como esa. Pero será una situación del mundo real en la que tendrás que pensar en cómo vas a obtener los datos que necesitas
- A menos que tengas mucha suerte y alguien ya tenga los datos por ti, lo cual es genial.

ML: Data is a first class citizen

- Software 1.0
  - Explicit instructions to the computer
- Software 2.0
  - Specify some goal on the behavior of a program
  - Find solution using optimization techniques
  - Good data is key for success
  - Code in Software = Data in ML



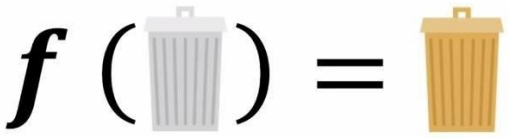
- En el diseño del lenguaje de programación, un ciudadano de primera clase es una entidad que soporta todas las operaciones generalmente disponibles para otras entidades. En ML, los datos son un ciudadano de primera clase.
- Así que en el software 1.0 todo era código, en realidad son las instrucciones para el ordenador
- En el software 2.0, necesitamos especificar un objetivo y el comportamiento del programa. El código es importante, pero no es lo único que nos preocupa,
- La optimización es realmente la fuerza motriz aquí, y puede ocurrir en muchas direcciones diferentes. Queremos optimizar el rendimiento, pero también queremos optimizar el mantenimiento y la escalabilidad. Y para el ML, la calidad de los datos es realmente crítica para el éxito, por lo que de alguna manera se podría ver como, los datos son casi como el código en un software

Everything starts with data

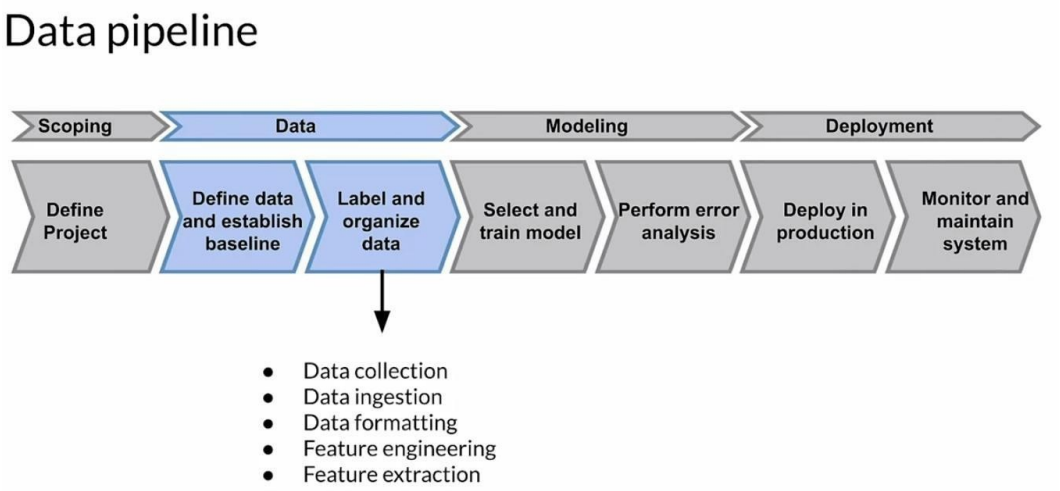
- Models aren't magic
- Meaningful data:
  - maximize predictive content
  - remove non-informative data
  - feature space coverage

- Es bueno que sepas que puedes tener montañas de datos, pero si no tienen contenido predictivo, no vas a poder crear un modelo predictivo con ellos.
- Es conveniente eliminar del modelo la información y las características que no son predictivas, porque van a causar problemas y, sin duda, van a consumir muchos recursos informáticos.
- Hay que asegurarse de que los datos de entrenamiento cubren el mismo espacio de características que las solicitudes de predicción, para que el modelo tenga buena información sobre las regiones de ese espacio para hacer predicciones

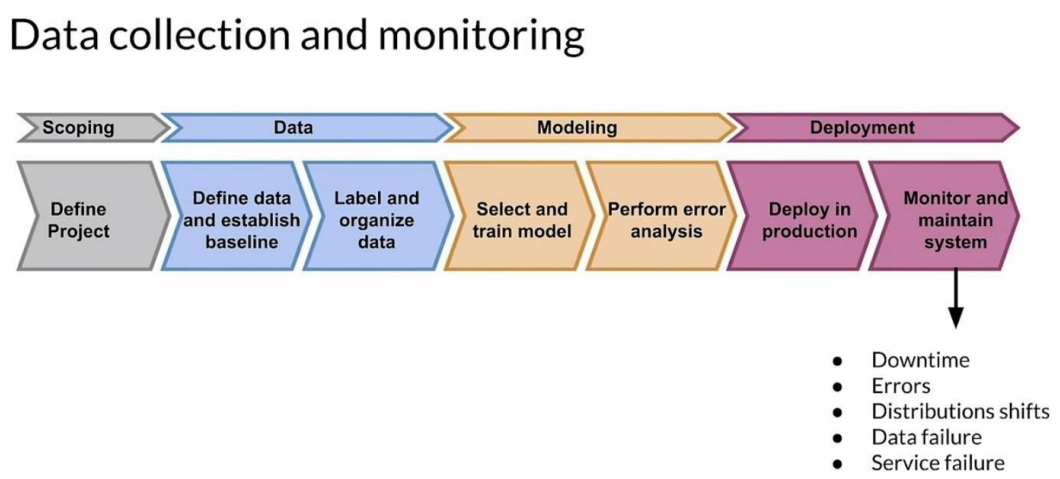
Garbage in, garbage out



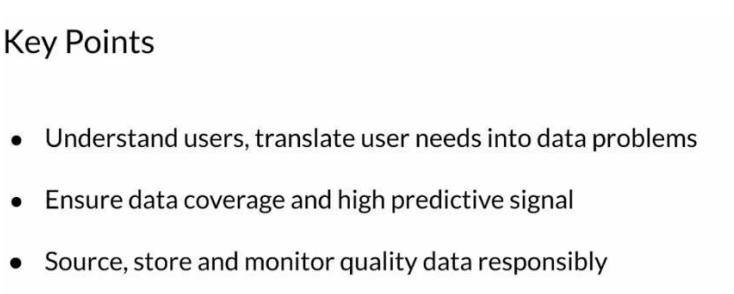
- Como todo, es basura de entrada y basura de salida. Así que si tus datos son basura, o si la calidad de tus datos es baja, tu modelo y tu aplicación serán de baja calidad.



- La recopilación de datos es un primer paso importante y crítico para la creación de sistemas de LD



- En el despliegue, usted quiere evitar problemas con el tiempo de inactividad
- Hay que asegurarse de que el modelo de formación puede escalar y servir para hacer predicciones
- Hay que pensar en los diferentes tipos de errores y en lo que vamos a hablar de todos ellos.
- Pero es todo este panorama el que hay que tener en cuenta a la hora de desarrollar todo el proceso, desde la ingesta hasta el servicio, y todo tiene que estar automatizado.
- Todo tiene que poder probarse y mantenerse y escalarse bien, etc., por lo que hay que entender a los usuarios y asegurarse de que se traducen las necesidades de los usuarios en problemas de datos.
- No se quiere hacer un modelo que no responda a las necesidades del usuario.



- En general, hay que asegurarse de que los datos cubren la misma región de su espacio de características que la solicitud de predicción para maximizar la señal de predicción en esos datos.

- Hay que preocuparse por la calidad de los datos no sólo al principio, sino durante toda la vida de la aplicación.
- Parte de esto es asegurarse de que se obtienen los datos de forma responsable y se piensa en cosas como la parcialidad y la imparcialidad

Ejemplo de aplicación - Sugerir recorridos

Example application: Suggesting runs

Users	Runners
User Need	Run more often
User Actions	Complete run using the app
ML System Output	• What routes to suggest • When to suggest them
ML System Learning	• Patterns of behaviour around accepting run prompts • Completing runs • Improving consistency

- Para este ejemplo, vamos a ver una aplicación que sugiere carreras a los corredores.
- Hay diferentes corredores con diferente nivel de forma física. El primer paso es realmente tratar de entender a los usuarios
- Este sistema va a sugerir recorridos basados en el comportamiento del usuario y aprovechando los patrones y preferencias observados.
- El objetivo es mejorar la consistencia de la carrera y que los corredores completen esas carreras y se sientan realmente felices por ello.

Key considerations

- Data availability and collection
  - What kind of/how much data is available?
  - How often does the new data come in?
  - Is it annotated?
    - If not, how hard/expensive is it to get it labeled?
- Translate user needs into data needs
  - Data needed
  - Features needed
  - Labels needed

- En primer lugar, hay que tener en cuenta la calidad de los datos y su recogida. ¿Qué tipo de datos y cuántos necesita? ¿Con qué frecuencia necesita nuevos datos? ¿Cuándo espera que las cosas cambien? ¿Están los datos anotados?
- Primero tenemos que entender al usuario, de lo contrario corremos el riesgo de recoger un montón de datos que en realidad son basura.
- Pero una vez que entendemos al usuario, tenemos que traducir las necesidades del usuario en necesidades de datos. Para ello vamos a identificar cuáles son los datos, cuáles son las características y cuáles son las etiquetas.

Example dataset

		FEATURES			
	Runner ID	Run	Runner Time	Elevation	Fun
EXAMPLES	AV3DE	Boston Marathon	03:40:32	1,300 ft	Low
	X8KGF	Seattle Oktoberfest 5k	00:35:40	0 ft	High
	BH9IU	Houston Half-marathon	02:01:18	200 ft	Medium

LABELS

- Aquí hay un conjunto de datos de ejemplo. Tenemos tres ejemplos diferentes aquí para tres tipos diferentes de carreras y tenemos algunas características.
- Los ejemplos son el maratón de Boston, la Oktoberfest 5K de Seattle y la media maratón de Houston.
- Las características son la propia carrera, el tiempo del corredor y la elevación, que también es importante.
- Entonces las etiquetas aquí van a ser simplemente cómo el corredor califica el nivel de diversión de esas carreras.

Get to know your data

- Identify data sources
- Check if they are refreshed
- Consistency for values, units, & data types
- Monitor outliers and errors

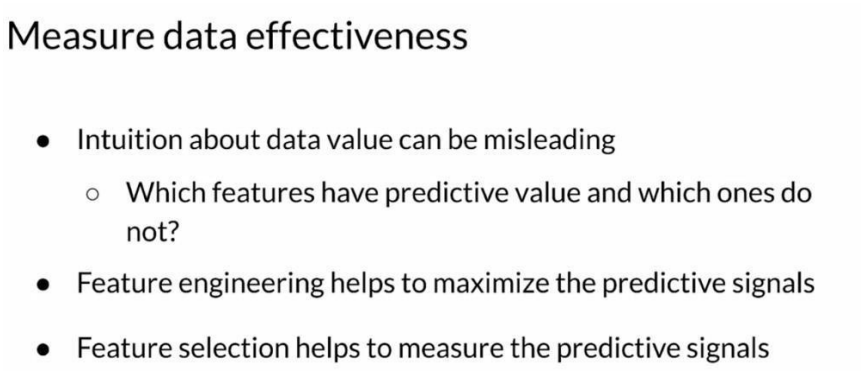
- Hay que identificar las fuentes de datos que se van a utilizar, dónde se van a obtener estos datos y no sólo la primera vez, sino de forma continuada
- No se trata sólo de la formación, sino que hay que recoger esos mismos datos para poder hacer una inferencia cuando se quiera crear una predicción.
- Tienes que pensar en la frecuencia con la que tengo que refrescar mis entrenamientos. A lo largo del proceso, cuando trabajes con tus datos, tienes que asegurarte de que realmente hay valor predictivo en tus datos. Asegúrate de que has eliminado las características y los datos que no tienen valor predictivo.
- También hay algunas cosas más básicas como, ¿son los datos consistentes? Cuando esperas, por ejemplo, un flotante, ¿siempre obtienes un flotante o es mixto? También hay que buscar cosas como los valores atípicos o los errores.

Dataset issues

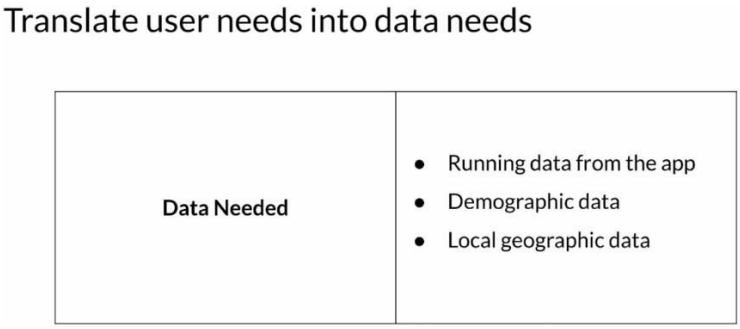
- Inconsistent formatting
  - Is zero “0”, “0.0”, or an indicator of a missing measurement
- Compounding errors from other ML Models
- Monitor data sources for system issues and outages

- Puede haber problemas con los datos debido a las diferentes medidas, a los diferentes tipos y también a cosas sencillas como la diferencia entre un int y un float, o cómo se codifican los valores que faltan, todo lo cual puede causar problemas.
- En este conjunto de datos de ejemplo, si la elevación es de cero pies, ¿significa eso realmente que estamos a nivel del mar o significa que no tenemos ningún dato de elevación para ese registro?

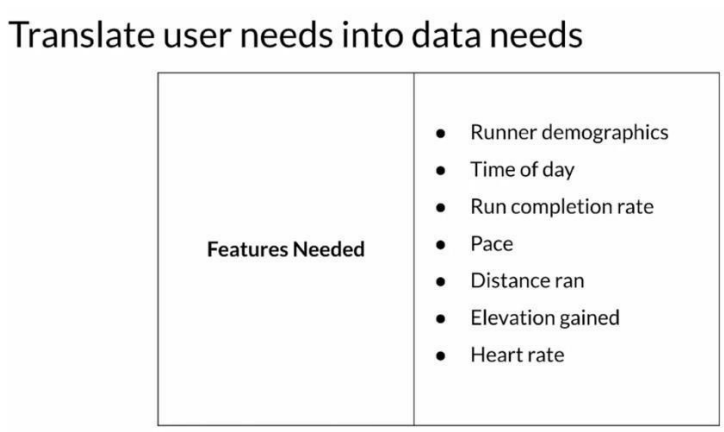
- Si la salida proviene de otros modelos de ML (tal vez estés usando un conjunto), hay errores en ellos que pueden agravarse cuando intentas usarlos en un modelo posterior.
- También hay que asegurarse de que se buscan errores y problemas en las primeras fases del proceso y de que se supervisan las fuentes de datos para detectar problemas y cortes del sistema. Porque podríamos estar operando esta cosa las 24 horas del día.



- Hay que tener cierta intuición sobre el valor de los datos, pero la intuición puede ser engañosa.
- Debes asegurarte de que estás mirando qué datos te están dando realmente la mayor información.
- La selección de características y la ingeniería de características son realmente críticas para dar forma a sus datos para que sean lo que usted necesita. La ingeniería de características te ayuda a maximizar las señales predictivas una vez que has identificado dónde están.
- La selección de características le ayuda a medir dónde se encuentra esa información predictiva y a centrarse en esas características para obtener el mayor valor y ayudar a su modelo.



- Hay que entender al usuario y la aplicación
- En este caso, estamos buscando datos de funcionamiento de una aplicación. Podemos obtener datos demográficos cuando el usuario rellena su perfil y probablemente también podemos obtener algunos datos de GPS para darnos alguna información geográfica local. A alto nivel, eso nos ayuda a entender al usuario.



- Luego hay que traducirlo en características.

- Los datos demográficos de los corredores, tenemos que expresarlos como una característica o probablemente varias características. Cosas como la hora del día, el tiempo que tardan en completar una carrera, su ritmo durante la carrera, la distancia, etc.
- También podemos obtener información sobre la elevación, y si estamos trabajando con una aplicación que tiene algunos sensores como un monitor de ritmo cardíaco, es una gran información para tener que alimentar realmente esta aplicación.

Translate user needs into data needs

Labels Needed	• Runner acceptance or rejection of app suggestions • User generated feedback regarding why suggestion was rejected • User rating of enjoyment of recommended runs

- La aceptación del corredor en este caso es una etiqueta en la que queremos centrarnos. Corredores que aceptan nuestras sugerencias y las utilizan. Eso nos dice que la aplicación les dio con éxito una carrera que querían hacer y, por el contrario, si la rechazaron.
- Comentarios generados por los usuarios. Tienes que pensar, en primer lugar, en cómo te van a dar ese feedback de forma estructurada que puedas utilizar para ayudar a entrenar tu modelo.
- Entonces sabes cosas como la valoración de los usuarios. En este caso, se refiere al disfrute de las carreras recomendadas.

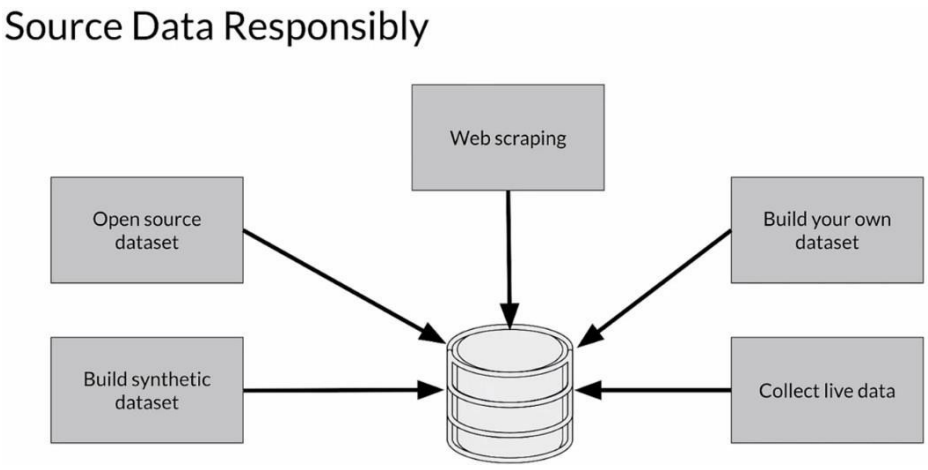
Key points

- Understand your user, translate their needs into data problems
  - What kind of/how much data is available
  - What are the details and issues of your data
  - What are your predictive features
  - What are the labels you are tracking
  - What are your metrics

- Puntos clave: entienda a su usuario y traduzca sus necesidades en problemas de datos y características bien definidas que le proporcionen información predictiva.
- Las preguntas que hay que hacerse son: ¿Qué tipo de datos puede obtener? ¿Qué hay disponible? ¿Cuáles son los detalles y problemas de sus datos? ¿Dónde está la información predictiva en tus datos? ¿Cuáles son las etiquetas? Tenemos que asegurarnos de que estamos entrenando un modelo para predecir lo correcto. Tenemos que asegurarnos de que nuestras etiquetas son las correctas para nuestros objetivos. ¿Cuáles son las métricas que debemos utilizar para medir el rendimiento de nuestro modelo?

Datos responsables: Seguridad, privacidad y equidad

- He aquí un ejemplo. Estas imágenes muestran un clasificador de imágenes estándar de código abierto entrenado en el conjunto de datos de imágenes abiertas que no aplica correctamente las etiquetas relacionadas con las bodas a las imágenes de las tradiciones nupciales de diferentes partes del mundo.
- En el extremo izquierdo, la predicción de la etiqueta del clasificador se registra como ceremonia, con las etiquetas boda, novia, hombre, grupo, mujer, vestido. Así que es bastante correcto.
- La siguiente es la ceremonia de la novia, la boda, el vestido y la mujer. De nuevo sabemos que es correcto, al menos en Occidente, que es el aspecto típico de una novia.
- Pero el del final, el de la derecha. Bueno, eso es para una ceremonia de boda africana, pero está incorrectamente etiquetado como simplemente persona o gente. Bueno, es persona y gente, pero también es una ceremonia y hay una novia y un novio y la dirección y así sucesivamente. Así que este es un caso clásico, es un ejemplo que se cita a menudo de un problema de sesgo en el conjunto de datos.



- En un sistema de inteligencia artificial, los datos pueden provenir de diferentes fuentes y también hay que pensar en ellas. No se trata solo de los datos que tienes, sino de dónde los has obtenido.
- Así que puedes construir datos sintéticos, puedes hacer scraping de la web o recoger datos en vivo, especialmente cuando estás ejecutando inferencia.
- Casi siempre vas a construir tu propio conjunto de datos, aunque a veces puedes utilizar un conjunto de datos de código abierto. Depende de lo que esté disponible y de lo que necesites.

Data security and privacy

- Data collection and management isn't just about your model
  - Give user control of what data can be collected
  - Is there a risk of inadvertently revealing user data?
- Compliance with regulations and policies (e.g. GDPR)

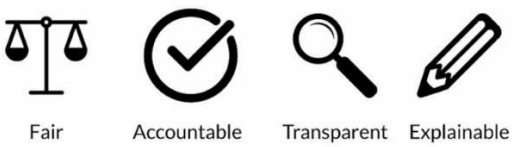
- La seguridad de los datos se refiere a las políticas y métodos para proteger los datos personales o lo que se suele denominar PII, Personally Identifiable Information.
- La privacidad de los datos tiene que ver con el uso adecuado, la recopilación, la retención, la eliminación y el almacenamiento de esos datos.
- Por lo tanto, la recopilación de datos no se limita a su modelo. Tienes que pensar en tus usuarios y tratar esos datos como algo que te han dado, del que eres un administrador responsable de gestionar esos datos de forma responsable.
- Los usuarios deberían tener control sobre los datos que se recogen. Y es importante establecer mecanismos para evitar que su sistema revele los datos de un usuario inadvertidamente
- La forma de gestionar la privacidad y la seguridad de los datos depende de la naturaleza de los mismos, así como de las condiciones de funcionamiento y de las normativas y políticas, siendo importantes aquí aspectos como el GDPR.

Users privacy

- Protect personally identifiable information
  - Aggregation - replace unique values with summary value
  - Redaction - remove some data to create less complete picture

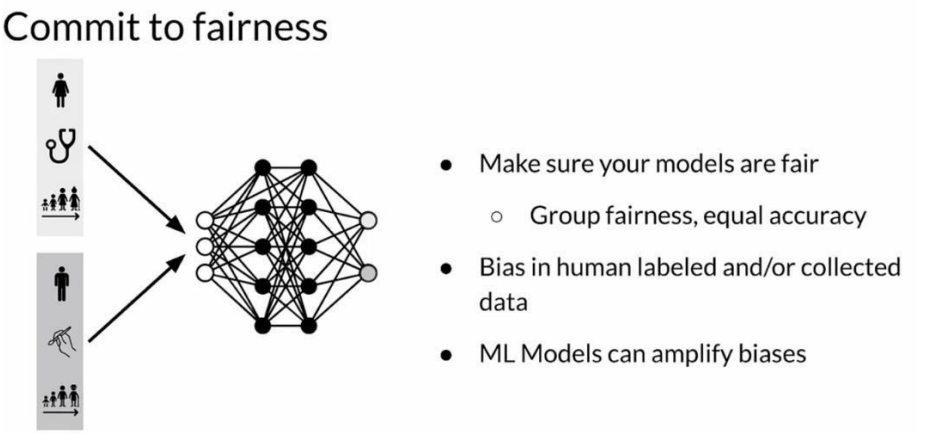
- La privacidad de los usuarios también es realmente clave. Así que hay que proteger la información o los datos de identificación personal
- La agregación ayuda mucho a ello, si se pueden agregar los datos de manera que no se pueda identificar a las personas individuales dentro de ellos
- Debes tener en cuenta las leyes o normativas relativas a la privacidad de los usuarios en los lugares donde vayas a utilizar tu modelo
- Otra forma es la redacción. En muchos casos hay que ofrecer a los usuarios la posibilidad de eliminar parte de los datos, lo que creará una imagen menos completa, pero forma parte de la responsabilidad con los datos.

How ML systems can fail users

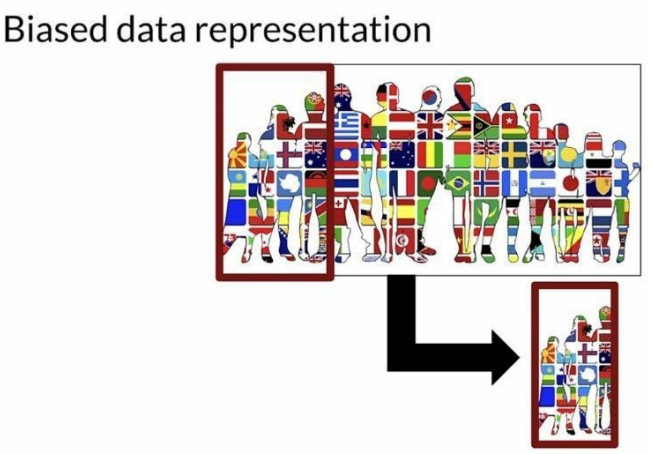


- Representational harm
- Opportunity denial
- Disproportionate product failure
- Harm by disadvantage

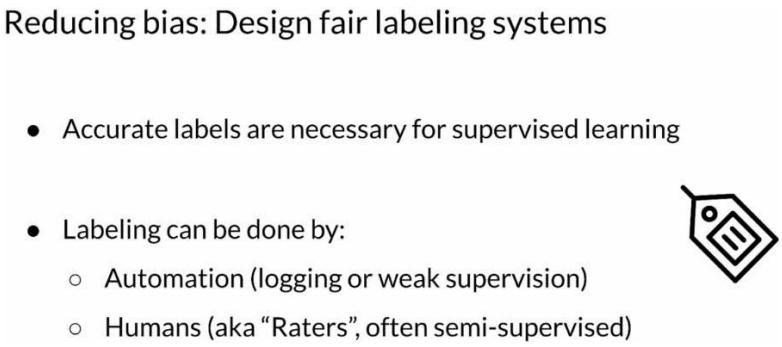
- Así que los sistemas de ML pueden fallar a los usuarios de muchas maneras diferentes y tenemos que encontrar un equilibrio entre ser justos y precisos y transparentes y explicables.
- Algunas de las formas en que los sistemas de LD pueden fallar son por cosas como el daño representacional.
- El daño de representación se produce cuando un sistema amplifica o refleja un estereotipo negativo sobre determinados grupos.
- La negación de la oportunidad se produce cuando un sistema hace predicciones que tienen consecuencias negativas en la vida real que podrían tener un impacto duradero.
- El fracaso desproporcionado del producto es cuando la eficacia de su modelo está realmente sesgada de modo que los resultados se producen con más frecuencia para determinados grupos de usuarios, se obtienen resultados sesgados con más frecuencia. Esencialmente se puede pensar en errores con mayor frecuencia.
- El daño por desventaja es cuando un sistema infiere asociaciones desventajosas entre diferentes características demográficas y los comportamientos del usuario en torno a eso.



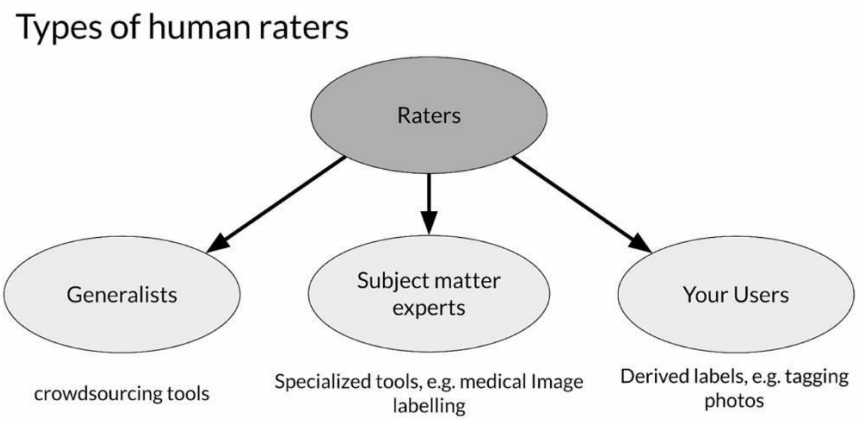
- Así que la equidad es importante y deberías comprometerte con ella desde el principio. ¿Y qué significa?
- Bien, ser justo significa que vas a identificar si algunos grupos de personas tienen una experiencia diferente a la de otros de forma problemática.
- Por ejemplo, supongamos que determinados campos de género, ocupación o edad forman parte de sus datos y los utiliza para entrenar un modelo que predice si alguien sería un nuevo empleado fiable. Entonces, interviene en la contratación.
- Debe comprobar realmente que su modelo no predice sistemáticamente experiencias diferentes para algunos grupos de forma problemática, garantizando la equidad del grupo.
- Lo que significa es la paridad demográfica y que las cosas se igualen entre los diferentes grupos.
- Y tienes que asegurarte de que la precisión también es igual o lo más cercana que puedas conseguir.
- Los datos recogidos y etiquetados por los humanos reflejarán sus prejuicios en muchos casos y sus experiencias personales, por lo que hay que tenerlo en cuenta.
- Diversificar la base de usuarios es una buena manera de avanzar hacia la equidad, pero no es una garantía.
- Los sistemas de ML pueden **amplificar** los sesgos, por lo que hay que ser consciente de ello y tener cuidado



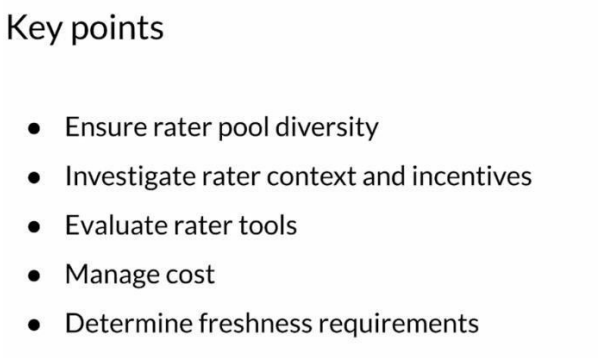
- Los sesgos también pueden surgir cuando hay una representación desproporcionada de algunos grupos en los datos o cuando no hay representación alguna.
- Así que mirando el gráfico aquí, lo que estamos tratando de mostrar es que parte de las personas que se muestran aquí están en sus datos, pero hay un montón de otras personas que no están.
- Esos grupos que podrían ser estereotipados son los que no están en sus datos, y podrían ser presentados de una manera menos positiva, o simplemente podrían tener una mala experiencia mucho más a menudo.



- Para reducir el sesgo del aprendizaje supervisado, se necesitan etiquetas precisas para entrenar el modelo y servir las predicciones.
- Las etiquetas suelen proceder de dos grandes fuentes. La mayoría de las veces provienen de sistemas automatizados o de calificadores humanos
- Los seres humanos son capaces de etiquetar los datos de diferentes maneras. Y cuanto más complicados sean los datos, más se necesitará un experto para analizarlos

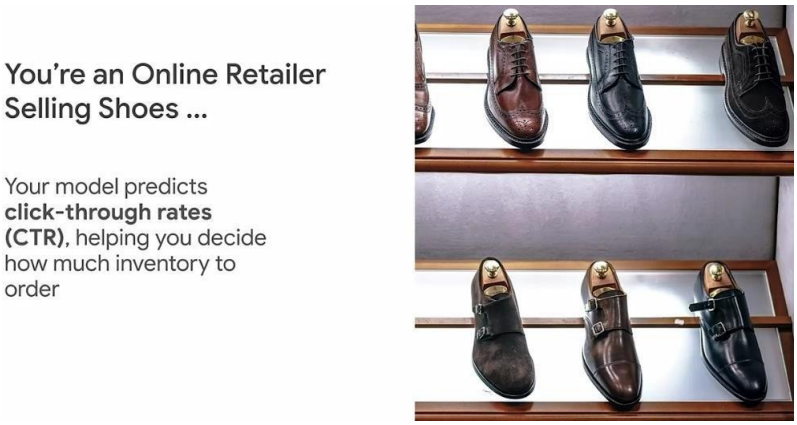


- ¿Quiénes son los calificadores? Bueno, podrían ser generalistas y se trata de gente bastante normal que va a añadir etiquetas a través de una variedad de herramientas de crowdsourcing. Y estos son casos en los que es bastante fácil que la gente reconozca la etiqueta correcta.
- Así, por ejemplo, si quieres que un humano reconozca la diferencia entre un gato y un perro, eso es algo que la mayoría de la gente puede hacer mirando la imagen.
- Pero en algunos casos se necesita realmente un experto en la materia. Así que en esos casos se suelen utilizar herramientas especializadas y un ejemplo de ello es el examen de las radiografías para el diagnóstico. No es algo que pueda hacer cualquiera. Tienes que asegurarte de que trabajas con un experto y el etiquetado tiende a ser bastante caro.
- También puedes utilizar a tus usuarios. Así que este tipo de retroalimentación que miramos para nuestra aplicación en funcionamiento. Esto a menudo puede ser muy valioso si usted puede encontrar una manera de trabajar sin en su aplicación y que va a dar este flujo continuo de etiquetas para sus datos si usted puede hacer que el trabajo.



- En primer lugar, tenga siempre en cuenta los calificadores y la representación equitativa en su conjunto de datos para evitar posibles sesgos.
- Tenga en cuenta quiénes son esos etiquetadores y cuáles son sus incentivos, porque si **diseña los incentivos de forma incorrecta, podría obtener mucha basura en sus datos.**
- El coste siempre va a ser una consideración importante. Así que si se puede encontrar una manera de hacerlo con un alto nivel de calidad pero a menor coste, es genial.
- Por último, la frescura de los datos también. Vas a trabajar con datos y, dependiendo de cómo cambie el contexto en torno a la aplicación y los datos que tienes, vas a necesitar refrescar esos datos con cierta regularidad y detectar cuándo necesitas hacerlo.

Estudio de caso - Rendimiento degradado del modelo



- Imagina que eres un vendedor online y que vendes zapatos. Tienes un modelo que predice las tasas de clics que te ayuda a decidir la cantidad de inventario que debes pedir.

- De repente, el AUC y la precisión de la predicción han caído, no en todo, sino en una parte concreta de su inventario, los zapatos de vestir para hombre. ¿Por qué?

Case study: taking action

- How to detect problems early on?
- What are the possible causes?
- What can be done to solve these?

- Desgraciadamente, si no se aplican buenas prácticas en el ámbito de la producción, es probable que se descubra cuando se pidan demasiados zapatos o que no haya suficientes. Y esa no es una situación en la que quieras estar en un negocio. Esto te va a costar dinero
- Así que hay que pensar en cómo se van a detectar problemas de este tipo a tiempo, y cuáles son las posibles causas para poder buscarlas y controlar el sistema.
- Y luego tratar de tener métodos y sistemas para hacer frente a esos problemas cuando ocurran, porque probablemente ocurrirán en algún momento.

What causes problems?

Kinds of problems:

- Slow - example: drift
- Fast - example: bad sensor, bad software update

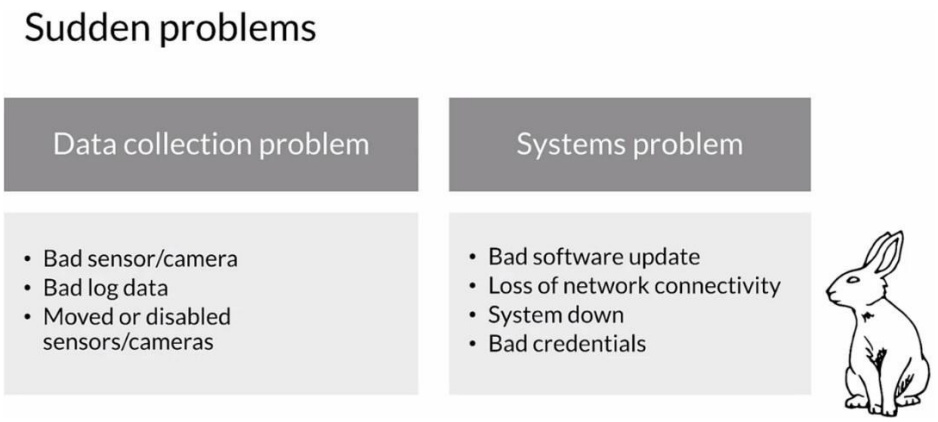
- ¿Pero qué tipo de problemas? Bueno, hay diferentes tipos y tienden a caer en dos categorías diferentes.
- Hay problemas de lentitud. Así, por ejemplo, tus datos se desviarán con el tiempo a medida que el mundo cambie y las estaciones pasen y tengas vacaciones y entren competidores, etc.
- Y luego tienes problemas rápidos que son realmente parte de tu sistema. Así que tienes un sensor que se estropea o tienes una actualización de software que se aplica y de repente las cosas están mal.
- Por lo tanto, hay que supervisar bien el sistema y buscar estos dos problemas y pensar en su solución en ambos casos.

Gradual problems

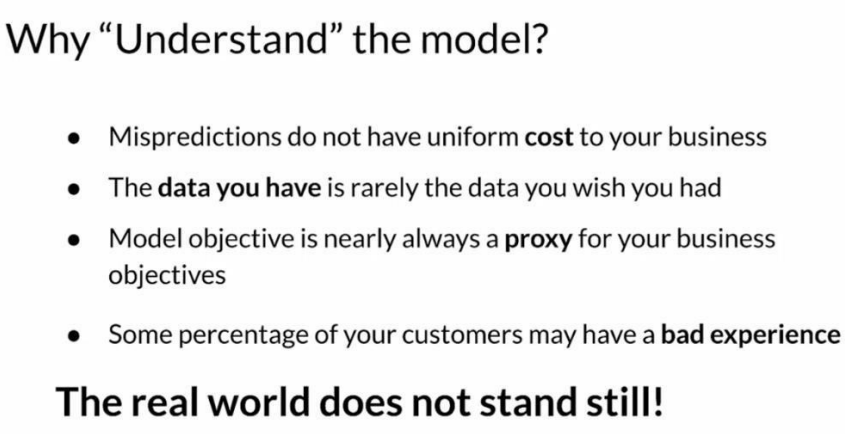
Data changes	• Trend and seasonality • Distribution of features changes • Relative importance of features changes
World changes	• Styles change • Scope and processes change • Competitors change • Business expands to other geos

- En el caso de los problemas graduales, tienden a caer en dos grupos, pero también están interrelacionados, así que no hay una línea dura entre ellos.
- La tendencia y la estacionalidad son un ejemplo, especialmente en las series temporales, donde habrá tendencias y estacionalidad en la mayoría de los casos. Se podría discutir si eso es realmente un cambio en los datos o un cambio en el mundo.
- Lo mismo ocurre con la distribución de las características. Y la importancia relativa de las características también puede cambiar. Si no has reentrenado tu modelo, la precisión empieza a decaer.

- El mundo también cambia constantemente. Y en los entornos de producción esto tiene que formar parte de los sistemas que se diseñan y de los procesos que se aplican.
- Por ejemplo, si trabajamos en el sector minorista y vemos un ejemplo con los zapatos, los estilos de moda cambian. Así, el año pasado los zapatos negros estaban muy de moda para los zapatos de vestir de los hombres y ahora son los zapatos marrones o lo que sea.
- El ámbito y los procesos cambian. Así que tu comprensión de esos procesos y de cómo ocurren en el mundo afectará a la forma en que tu modelo ve los resultados de esos procesos.
- Los competidores cambian y su negocio también cambia. Por lo tanto, es posible que adquiera nuevos productos, o que deje de vender algunos otros.
- Esto tiende a ser muy específico del dominio, pero en general, en casi todos los dominios, los cambios en el mundo afectan al rendimiento de su modelo.



- Para los problemas repentinos, hay cosas con las que probablemente estés familiarizado. En los problemas de recogida de datos, cosas como un sensor malo o una cámara mala, los datos de registro cambian de repente y tienes un formato diferente.
- En cuanto a los problemas de los sistemas, vas a tener actualizaciones de software todo el tiempo y si cambian algo importante de lo que no eras consciente, eso puede ser un verdadero problema.
- Conectividad de la red. A veces la red o el sistema se caen, otras veces sí.
- Y las credenciales incorrectas, como por ejemplo si sus credenciales de inicio de sesión caducan o algo cambia en ellas, todo eso puede causar un problema repentino y hay que solucionarlo.



- Es importante entender su modelo y cómo es sensible a los diferentes cambios en el mundo.
- Un problema importante es que los errores de predicción no tienen un coste uniforme para su empresa. Algunas predicciones erróneas tendrán muy poco efecto en su negocio. Otras predicciones erróneas podrían tener efectos enormes.
- Por lo tanto, es importante comprenderlo y, al supervisar las cosas, buscar las que tienen un mayor impacto.
- Los datos que tienes mientras recoges los datos rara vez son los que desearías tener. Personalmente, he visto casos en los que utilizábamos datos de sensores de dispositivos wifi y no eran datos muy buenos. Había mucho ruido, pero era todo lo que teníamos. Así que tuvimos que trabajar con lo que teníamos.

- El objetivo del modelo es casi siempre una aproximación a lo que realmente se quiere conseguir. En muchos casos, puedes diseñar un modelo y tienes los datos para diseñar un modelo para exactamente lo que estás tratando de hacer.
- Pero a menudo, como en el caso del ejemplo de los zapatos que acabamos de ver, estábamos prediciendo las tasas de clics como una aproximación para decidir cuánto inventario pedir.
- Un porcentaje de sus clientes tendrá una mala experiencia. Usted quiere que ese porcentaje sea lo más pequeño posible
- Hay que saber qué clientes van a ser esos, para intentar diseñar formas de mitigarlo y mejorar la situación de todos los clientes.
- Pero la conclusión a la que te enfrentarás constantemente es que el mundo real no se queda quieto. La única constante en el mundo es el cambio.

Cambio de datos y conceptos en el ML de producción

Detecting problems with deployed models

- Data and scope changes
- Monitor models and validate data to find problems early
- Changing ground truth: **label** new training data

- Hay que supervisar los modelos y validar los resultados y los datos de los mismos para encontrar problemas.
- Hay que tratar de encontrar los problemas en una fase temprana, especialmente cuando se trata de problemas del sistema que ocurren rápidamente, como un sensor defectuoso o cosas por el estilo.
- Pero un problema fundamental es el cambio de la verdad sobre el terreno. Esto significa que hay que etiquetar nuevos datos a lo largo de la vida de la aplicación. Depende del ámbito en el que trabajes y del tipo de problemas que intentes resolver

Easy problems

- Ground truth changes slowly (months, years)
- Model retraining driven by:
  - Model improvements, better data
  - Changes in software and/or systems
- Labeling
  - Curated datasets
  - Crowd-based

- Hay problemas fáciles. Cosas como intentar reconocer imágenes de gatos y perros.
- En este caso, la verdad sobre el terreno cambia muy lentamente. El reentrenamiento del modelo en esos casos suele estar impulsado por las mejoras del modelo
- También podría haber cambios en el software. Podrías estar actualizando cosas o usando una biblioteca diferente, ese tipo de cosas, o sistemas.
- El etiquetado en este caso es bastante simple, vas a trabajar con un conjunto de datos curados que obtienes de alguna fuente de dominio público, o una fuente que tu organización ha estado utilizando durante un tiempo
- También se pueden buscar datos basados en la multitud.

Harder problems

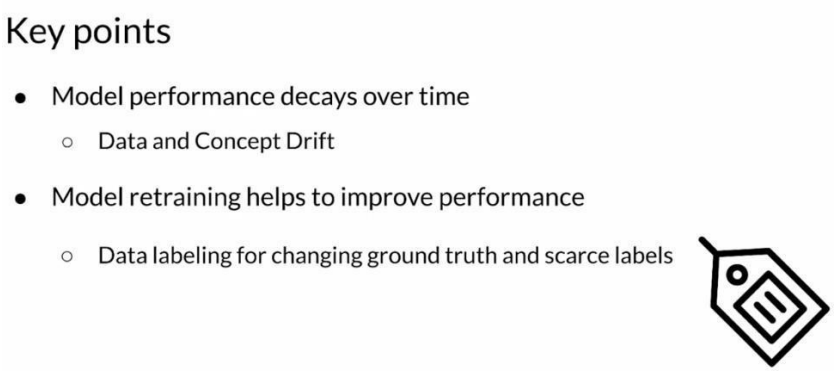
- Ground truth changes faster (weeks)
- Model retraining driven by:
  - **Declining model performance**
  - Model improvements, better data
  - Changes in software and/or system
- Labeling
  - Direct feedback
  - Crowd-based

- Luego nos metemos en problemas un poco más difíciles en los que la verdad del terreno cambia más rápido.
- Cosas como los estilos (por ejemplo, los zapatos), pero hay muchas cosas en el mundo que cambian en cuestión de tal vez semanas.
- El reentrenamiento del modelo en ese caso suele estar motivado por la disminución del rendimiento del modelo, que hay que medir si se va a tener en cuenta.
- También puede haber mejoras en los modelos que hay que aplicar, o también puede haber mejores datos y, por supuesto, el software y los sistemas que se utilizan también pueden cambiar.
- Para el etiquetado, si puedes obtener información directa de tus sistemas o de tus usuarios, es genial.
- El etiquetado humano basado en la multitud es otra forma factible de hacerlo, ya que probablemente tenga semanas para responder. Puede pasar por calificadores humanos para hacerlo.

Really hard problems

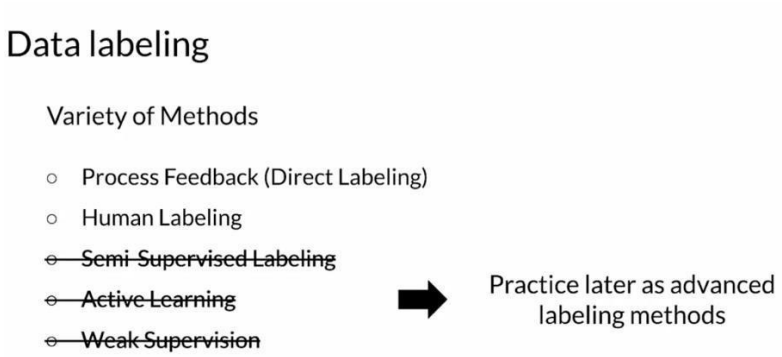
- Ground truth changes very fast (days, hours, min)
- Model retraining driven by:
  - **Declining model performance**
  - Model improvements, better data
  - Changes in software and/or system
- Labeling
  - Direct feedback
  - Weak supervision

- En este caso, la verdad sobre el terreno cambia muy rápidamente, como en el orden de días u horas o incluso minutos.
- Cosas como los mercados de valores entran en esta categoría, cambian muy rápidamente.
- En este caso, la disminución del rendimiento del modelo será sin duda un factor determinante a la hora de tener que volver a entrenar el modelo.
- También puede haber cosas como mejoras en el modelo y cambios en el software, etc., pero suelen ser cosas en las que se trabaja fuera de línea mientras se mantiene la aplicación en funcionamiento.
- Es realmente el rendimiento del modelo donde se necesitan procesos bien definidos para hacer frente a esos cambios.
- El etiquetado en este caso se convierte en un gran reto. La retroalimentación directa es genial si se puede hacer en su dominio.
- Si no es así, hay que ver cosas como la supervisión semanal de la que hablaremos. Pero es realmente un reto en este tipo de dominios. Tienden a ser dominios de alto valor. Cosas como la predicción de los mercados donde hay un incentivo significativo para hacer estas predicciones.

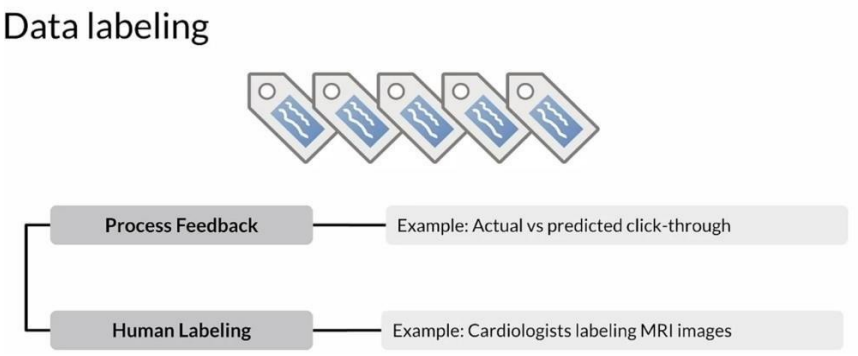


- Los puntos clave de lo que estamos hablando aquí, incluye el punto de que el rendimiento del modelo decae con el tiempo. Puede decaer lentamente con el tiempo, en cosas como perros y gatos, que no cambian muy rápidamente, o puede cambiar muy rápido, cosas como los mercados.
- El reentrenamiento del modelo le ayudará a mejorar o mantener su rendimiento.
- El etiquetado de los datos, suponiendo que se haga un aprendizaje supervisado, lo que es bastante común, es una parte clave de eso. Hay que pensar realmente en cómo se va a enfocar eso en el problema concreto, en el dominio concreto y con los sistemas de los que se dispone.

Retroalimentación del proceso y etiquetado humano



- Existen numerosas formas de generar etiquetas para sus datos. Nos centraremos en las dos primeras, que son las más comunes, la retroalimentación del proceso o etiquetado directo y el etiquetado humano



- Se necesitan etiquetas si se va a realizar un aprendizaje supervisado
- Dos formas sencillas de hacerlo son la retroalimentación procesada y el etiquetado humano.
- Veamos algunos ejemplos, para la retroalimentación del proceso, un ejemplo muy típico es la tasa de clics. Los índices de clics reales frente a los previstos.
- Supongamos que tienes recomendaciones que le das a un usuario, ¿hizo realmente clic en las cosas que recomiendas? Si lo hicieron, puedes etiquetarlo como positivo, si no lo hicieron puedes etiquetarlo como negativo.

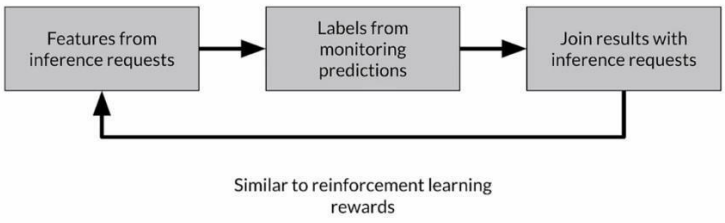
- Etiquetado humano, puede hacer que los humanos miren los datos y les apliquen etiquetas.
- Por ejemplo, puede pedir a los cardiólogos que observen las imágenes de la resonancia magnética y les apliquen etiquetas

Why is labeling important in production ML?

- Using business/organisation available data
- Frequent model retraining
- Labeling ongoing and critical process
- Creating a training datasets requires labels

- ¿Por qué es importante el etiquetado en el ML de producción? La mayoría de las empresas y organizaciones tienen un montón de datos, pero si no están etiquetados, no se pueden utilizar para el aprendizaje supervisado.
- Si puedes aplicar técnicas no supervisadas y obtener buenos resultados, es estupendo. Pero en muchos casos se necesita realmente el aprendizaje supervisado para resolver los problemas que se pretenden solucionar.
- Lo que significa que en la mayoría de los ámbitos vas a tener que volver a formarte en algún momento. Dependerá, como hemos dicho, del ámbito en el que trabajes y del tipo de problema.
- Algunos sólo necesitarán volver a entrenar con poca frecuencia, y otros podrían necesitar volver a entrenar varias veces al día.
- El etiquetado es un proceso continuo y a menudo crítico en su aplicación y su negocio

Direct labeling: continuous creation of training dataset

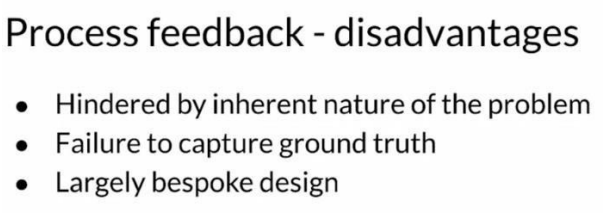


- El etiquetado directo es una forma de crear continuamente nuevos datos de entrenamiento para utilizarlos en el reentrenamiento del modelo.
- Estás tomando las características mismas de las solicitudes de inferencia que tu modelo está recibiendo. Las predicciones que se le piden a tu modelo y las características que se proporcionan para ello.
- Se obtienen etiquetas para esas solicitudes de inferencia mediante sistemas de supervisión y se utiliza la información de esos sistemas para etiquetar esos datos.
- Una de las cosas que hay que resolver es unir los resultados que se obtienen de la monitorización de esos sistemas con la solicitud de inferencia original, que podría estar separada por horas o días.
- Es posible que haya ejecutado lotes el lunes y que reciba información el viernes. Tienes que asegurarte de que puedes hacer esas uniones para aplicar esas etiquetas.
- En cierto modo, se puede pensar en esto como algo similar al aprendizaje por refuerzo, donde en lugar de aplicar recompensas basadas en la acción, se aplican etiquetas basadas en una predicción. Es un bucle de retroalimentación similar.

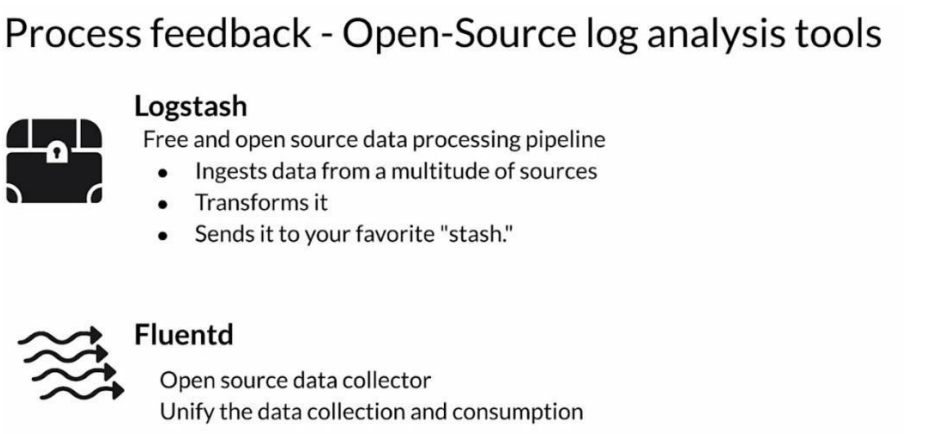
Process feedback - advantages

- Training dataset continuous creation
- Labels evolve quickly
- Captures strong label signals

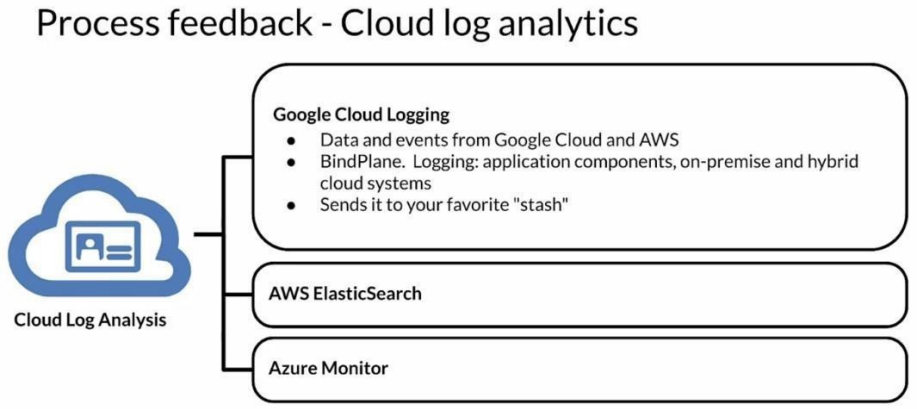
- Las ventajas del etiquetado directo son grandes, si tu sistema y tu dominio están configurados de forma que puedas hacerlo. A menudo es la mejor respuesta, ya que obtienes etiquetas de vuelta que estás monitoreando, y estás constantemente obteniendo nuevos datos de entrenamiento.
- Las señales que obtienes de tus etiquetas son realmente fuertes. Obtienes cosas como los clics, si el usuario hizo o no hizo clic, es una señal muy fuerte.



- En muchos ámbitos y para muchos problemas, desgraciadamente el etiquetado directo no es posible.
- Personalmente, en los problemas que me han pedido que resuelva, he encontrado muy pocos en los que haya podido hacerlo, así que eso es un problema.
- La otra gran cosa es que tiende a ser un diseño muy personalizado. Sus sistemas serán únicos, pero sería estupendo que estuvieran un poco más lejos de la media.

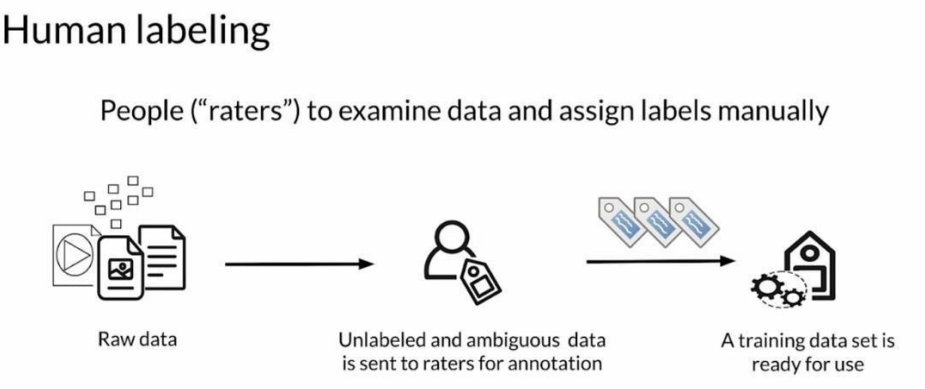


- Una de las herramientas que puedes aplicar son las de análisis de registros. Porque a menudo, cuando se hace la retroalimentación del proceso, los datos provienen de los archivos de registro. Estás monitoreando los sistemas y llenando los archivos de registro.
- Una buena herramienta de código abierto para hacerlo es **Logstash**. Puedes ingerir desde múltiples fuentes para recoger, analizar y almacenar los registros. Puedes indexarlos en Elasticsearch, y puedes empujarlos al almacenamiento.
- Toma datos de diferentes fuentes y bases de datos, etc.
- Gran herramienta y además es de código abierto.
- **Fluentd** es otra buena herramienta de código abierto que puede utilizar para recopilar y analizar. Fluentd proviene de la Cloud Native Computing Foundation y se conecta a un montón de plataformas diferentes.

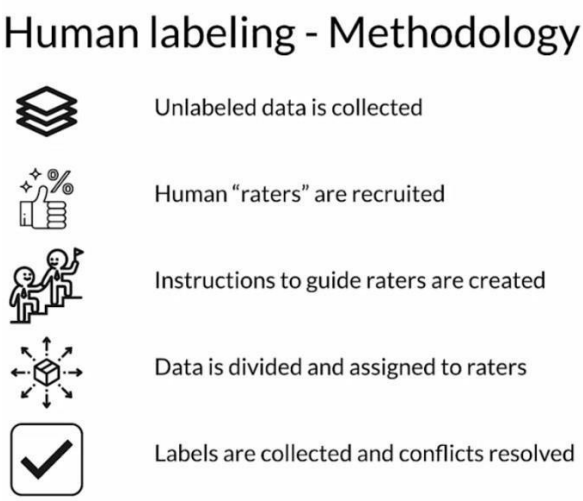


- Cuando trabajas en la Nube, también hay herramientas de la Nube que están disponibles. Si trabajas en la nube de Google, el registro de la nube de Google es una gran herramienta para poder registrar tus datos, ya sea que vengan de la nube de Google o de AWS.
- BindPlane es ideal para aplicar sistemas en las instalaciones o en la nube híbrida. Es un servicio muy potente.

- Para AWS, su versión de Elasticsearch está disponible, así que puedes aplicarla. No es estrictamente una herramienta de análisis de registros, pero se puede aplicar para el análisis de inicio de sesión.
- En el caso de Azure, tienes los monitores de Azure. Así que, independientemente de la nube en la que estés trabajando, probablemente haya herramientas de análisis de registros que puedas usar para hacer análisis de registros.

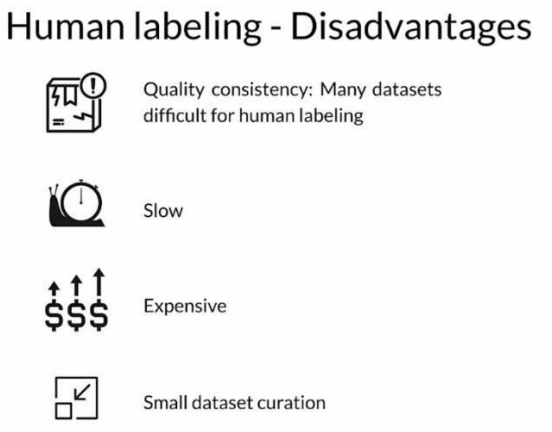


- En el etiquetado humano, hay personas, humanos, a los que llamamos calificadores.
- Les pedimos que examinen los datos y les asignen etiquetas.
- Empiezas con datos brutos y se los das a la gente y les pides que les apliquen etiquetas. Esa es la forma de crear un conjunto de datos de entrenamiento que vas a utilizar para entrenar o reentrenar tu modelo.

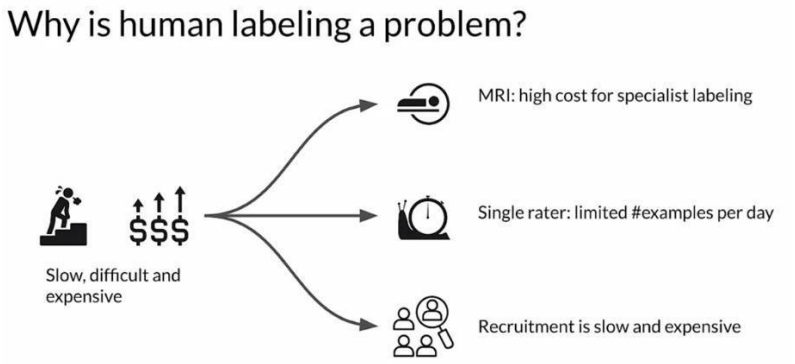


- Has empezado con datos no etiquetados y luego necesitas reclutar calificadores humanos
- Hay varios servicios a los que puedes acudir y que tienen grupos de calificadores humanos ya reclutados.
- Hay que darles instrucciones. Aunque sea muy sencillo, hay que decirles qué etiquetas deben aplicar y en qué deben fijarse para decidir qué etiqueta aplicar.
- A continuación, los datos se reparten entre los distintos calificadores del grupo. A menudo se envían los mismos ejemplos a varios calificadores, de modo que cuando hay desacuerdos, se es consciente de ello y se puede trabajar para resolverlos.
- Luego se recogen los datos y se resuelven los conflictos que se tengan.

- Ventajas del etiquetado humano. Permite generar las etiquetas que se necesitan para el aprendizaje supervisado. Es una forma de hacerlo y de hecho es una forma muy común de hacerlo.

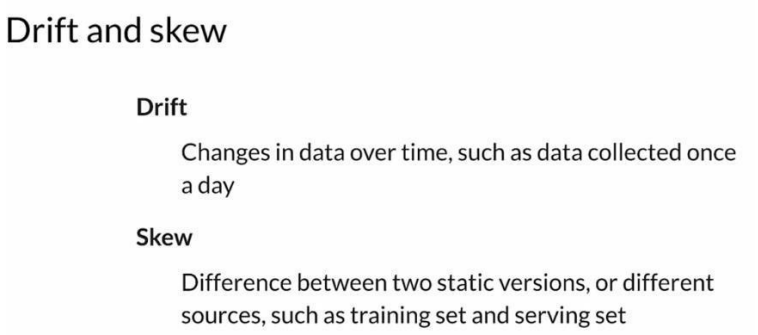


- Puede haber problemas de calidad cuando diferentes seres humanos no están de acuerdo en lo que debe ser la etiqueta.
- Puede ser muy lento. Le pides a la gente que mire cada ejemplo individual y puede llevar tiempo hacerlo
- En los casos en los que los datos cambian rápidamente, a menudo es simplemente inviable. Una vez más, puede ser muy costoso incluso si se utilizan generalistas para examinar datos muy simples.
- En los casos en los que se pide a los expertos que examinen los datos, entonces resulta muy caro.
- A menudo esto significa que se acaban teniendo conjuntos de datos pequeños porque el coste y el tiempo que conlleva dificultan la obtención de un conjunto de datos muy grande.



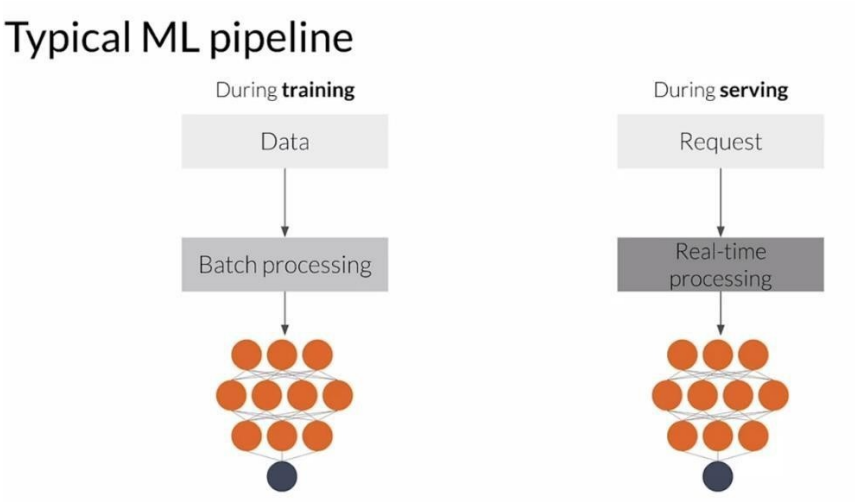
- En general, puede ser lento, difícil y caro.
- Si se hace algo como una resonancia magnética y un especialista la examina de nuevo, el coste es un problema.
- Un solo evaluador sólo puede realizar un determinado número de ejemplos al día, por lo que es necesario tener un grupo bastante grande en comparación con el número de ejemplos que se intenta etiquetar.

Detección de problemas de datos

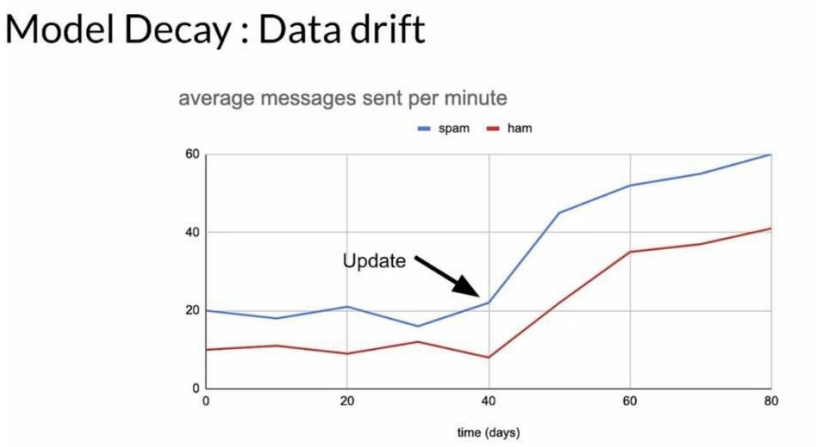


- La deriva son los cambios en los datos a lo largo del tiempo. Por ejemplo, los datos recogidos una vez al día en el tiempo, tal vez una semana más tarde, un mes más tarde, hay cambios que los datos han derivado.

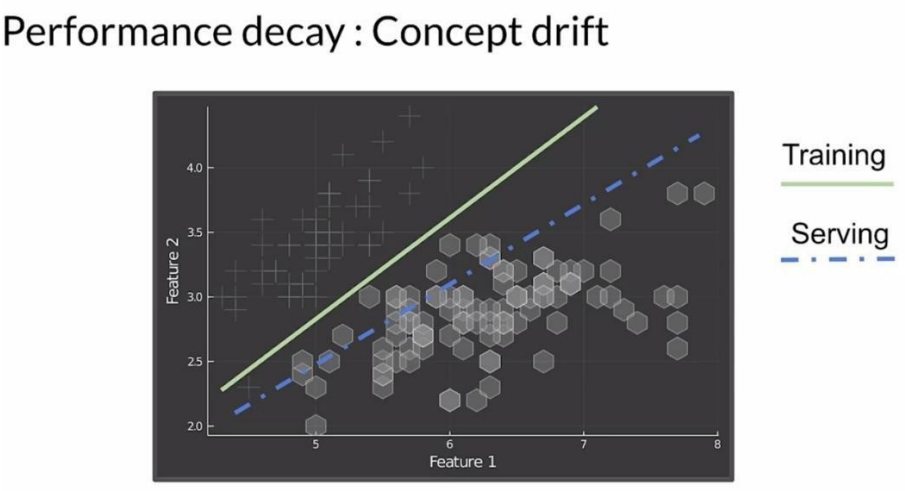
- La desviación es la diferencia entre dos versiones estáticas de diferentes fuentes del **mismo** conjunto de datos.
- Por ejemplo, podría ser la diferencia entre el conjunto de entrenamiento y los datos que se obtienen para las solicitudes de predicción, el conjunto de servicio. Esas diferencias se denominan sesgo.



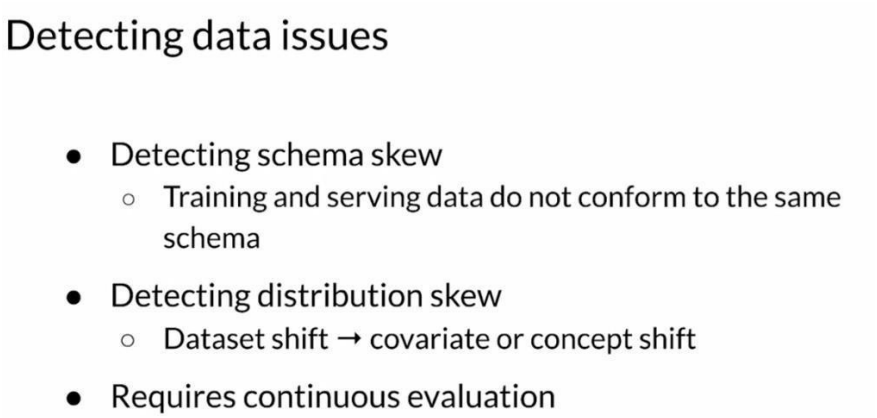
- En una tubería típica de ML, tenemos muestra de procesamiento por lotes, pero también podría ser el procesamiento en línea.
- Tienen el mismo vector de características, pero con el tiempo cambiarán.
- Esto significa que el rendimiento del modelo puede caer rápidamente debido a cosas como un fallo del sistema, o puede decaer con el tiempo debido a los cambios en los datos y los cambios en el mundo.
- Vamos a centrarnos en el deterioro del rendimiento que surge debido a los problemas entre los datos de entrenamiento y los de servicio.



- Existe la deriva de datos, que son los cambios en los datos entre el entrenamiento y el servicio típico y la deriva de conceptos, que son los cambios en el mundo cambios en la verdad del terreno.
- En el decaimiento del modelo a lo largo del tiempo, un modelo ML empezará a funcionar mal en muchos casos y nos referimos a esto como decaimiento del modelo. Eso suele estar causado por la deriva, que son cambios de forma conceptual.
- Hay cambios en las propiedades estadísticas de las características, o a veces debido a cosas como la estacionalidad o la tendencia o eventos inesperados.
- En este ejemplo, estamos viendo una aplicación de clasificación de spam
- Tras una actualización del sistema, tanto los spammers como los no spammers empiezan a enviar más mensajes.
- En este caso, los datos y el mundo han cambiado, y eso provoca una clasificación errónea no deseada



- La deriva de conceptos es un cambio en las propiedades estadísticas de las etiquetas a lo largo del tiempo.
- En el entrenamiento, un modelo de ML aprende un mapeo entre las características y las etiquetas. En un mundo estático eso está bien, no va a cambiar.
- Pero en el mundo real, la distribución y el significado de las etiquetas cambiarán. El modelo también tiene que cambiar, ya que el mapeo encontrado durante el entrenamiento ya no será válido



- Como se ha visto anteriormente, hay muchos factores que provocan cambios a lo largo del tiempo, como los cambios de los datos anteriores y la estacionalidad y la evolución de los procesos empresariales.
- La desviación del esquema se produce cuando los datos de entrenamiento y los datos de servicio no se ajustan al mismo esquema, lo que podría pensarse que nunca podría suceder, pero en realidad puede ocurrir porque se están recopilando datos y las cosas cambian y de repente se obtiene un entero donde se espera un flotante (es decir, los cambios de esquema).
- O se obtiene una cadena donde se espera una categoría.
- El sesgo de las distribuciones es una divergencia de los conjuntos de datos de entrenamiento y de servicio.
- El cambio de conjunto de datos puede manifestarse realmente por covariante y concepto y otros tipos de cambios. Hablaremos de eso en un segundo.
- La detección de sesgos implica la evaluación continua de los datos que llegan a su servidor una vez que entrena su modelo. Para detectar estos cambios, es necesario un seguimiento y una evaluación continuos de los datos.

Detecting distribution skew

	Training	Serving
Joint	$P_{\text{train}}(y, x)$	$P_{\text{serve}}(y, x)$
Conditional	$P_{\text{train}}(y\|x)$	$P_{\text{serve}}(y\|x)$
Marginal	$P_{\text{train}}(x)$	$P_{\text{serve}}(x)$

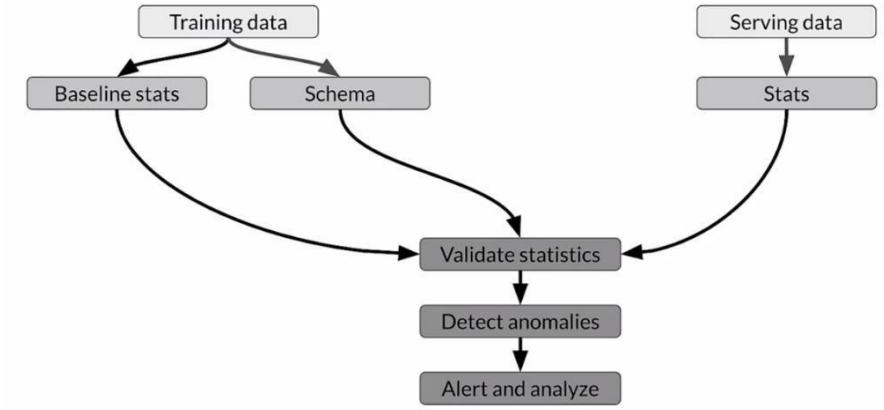
Dataset shift $P_{\text{train}}(y, x) \neq P_{\text{serve}}(y, x)$

Covariate shift $P_{\text{train}}(y|x) = P_{\text{serve}}(y|x)$, $P_{\text{train}}(x) \neq P_{\text{serve}}(x)$

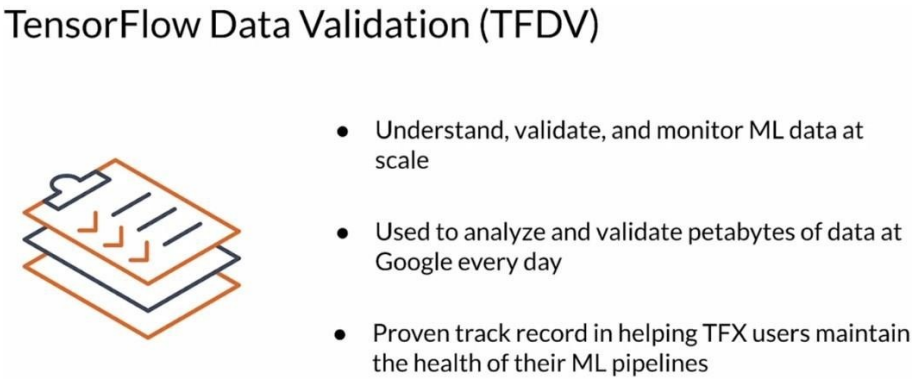
Concept shift $P_{\text{train}}(y|x) \neq P_{\text{serve}}(y|x)$, $P_{\text{train}}(x) = P_{\text{serve}}(x)$

- Veamos una definición más rigurosa de la deriva y la inclinación de la que hablamos.
- **El desplazamiento del conjunto de datos** se produce cuando la probabilidad conjunta de x (características) e y (etiquetas) no es la misma durante el entrenamiento y el servicio. Los datos se han desplazado con el tiempo.
- **El cambio de covariable** se refiere al cambio en la distribución de las variables de entrada presentes en los datos de entrenamiento y de servicio. En otras palabras, es cuando la distribución marginal de x (características) no es la misma durante el entrenamiento y el servicio, pero la distribución condicional no cambia.
- **El cambio de concepto** se refiere a un cambio en la relación entre las variables de entrada y salida, en contraposición a las diferencias en la distribución de datos o la entrada en sí. En otras palabras, es cuando la distribución condicional de y (etiquetas) dada x (características) no es la misma durante el entrenamiento y el servicio, pero la distribución marginal de x (características) no cambia.

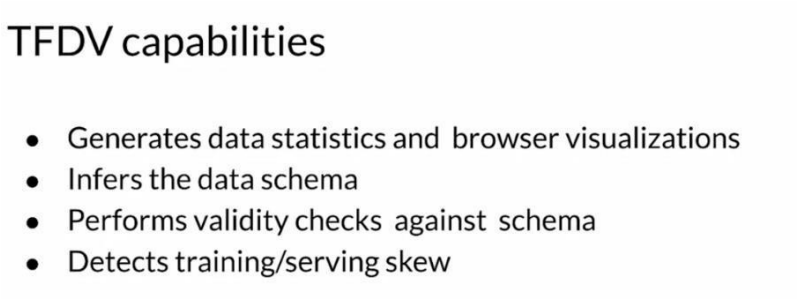
Skew detection workflow



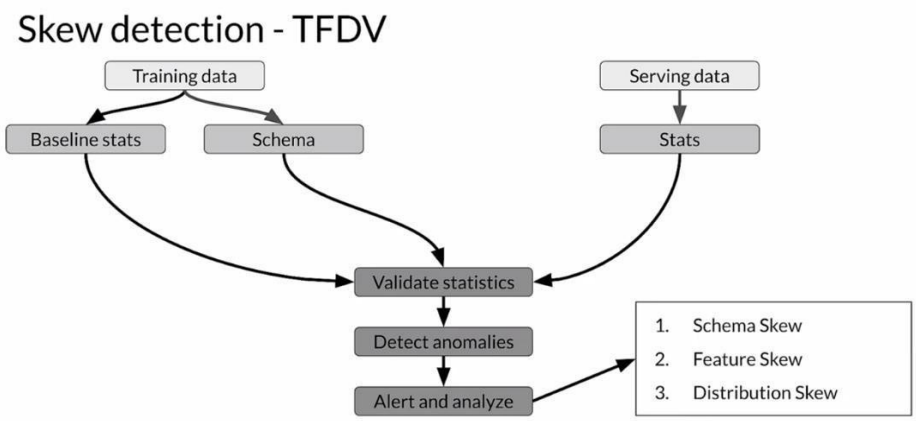
- Hay un flujo de trabajo sencillo para detectar la inclinación de los datos
- La primera etapa consiste en examinar los datos de formación y calcular las estadísticas de referencia y el esquema de referencia
- A continuación, haces básicamente lo mismo con tus datos de servicio, donde vas a generar las estadísticas descriptivas
- A continuación, se comparan los dos y se comprueban las diferencias entre el conjunto de datos de servicio y los datos de entrenamiento, es decir, se busca el sesgo y la deriva
- Los cambios significativos se convierten en anomalías y activan una alerta. Esa alerta se envía a quien esté supervisando el sistema, que puede ser un humano u otro sistema, para que analice el cambio y decida el curso de acción adecuado. Tiene que haber un plan de remediación en el que vas a arreglar y reaccionar a ese problema



- Ahora que has visto algunos de los problemas de datos y flujos de trabajo de detección y tienes una idea de la importancia de la validación de datos a escala de producción, vamos a echar un vistazo a la Validación de Datos de TensorFlow, que es una biblioteca de Google como parte del Ecosistema TFX.
- Te permitirá hacer validación de datos usando Python y haremos un ejercicio haciendo eso.
- TensorFlow Data Validation (**TFDV**) ayuda a los desarrolladores a comprender, validar y supervisar los datos de ML a escala.
- TFDV se utiliza para analizar y validar petabytes de datos en Google cada día en cientos o miles de aplicaciones diferentes que están actualmente en producción.
- TFDV ayuda a los usuarios de TFX a mantener la salud de sus pipelines de ML.



- TFDV ayuda a generar estadísticas de datos y proporciona visualizaciones en el navegador.
- También ayuda a deducir el esquema de sus datos, pero tendrá que asegurarse de que ese esquema tiene sentido.
- Una vez que tiene estas estadísticas y el esquema, puede buscar problemas son anomalías en sus datos.
- A continuación, se examinará el sesgo de servicio de entrenamiento comparando los datos de los conjuntos de datos de entrenamiento y de servicio.
- Uno de los casos de uso más comunes es la comprobación continua de los datos recién llegados mediante su validación con respecto a las expectativas que se tienen en el esquema de referencia que se ha generado a partir de los datos de entrenamiento.
- La configuración típica utiliza el esquema, que se mantiene en el tiempo, y las estadísticas se calculan sobre los nuevos datos. Esas estadísticas se utilizan luego para validar los datos con respecto al esquema original.

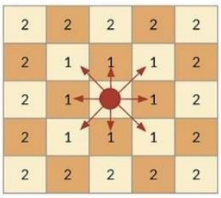


- Recuerde, hablamos de la detección de sesgo y con TFDV puede detectar fácilmente tres tipos diferentes de sesgo, sesgo de esquema, sesgo de característica y sesgo de distribución.

Skew - TFDV

- Supported for categorical features
- Expressed in terms of L-infinity distance (Chebyshev Distance):

$$D_{\text{Chebyshev}}(x, y) = \max_i(|x_i - y_i|)$$



- Set a threshold to receive warnings

- El TFDV realiza la detección de sesgos o desviaciones en características categóricas
- La inclinación se expresa en términos de una distancia **L-infinita,** que también se conoce como distancia de **Chebyshev**.
- Si piensas en un tablero de ajedrez, la métrica de la distancia es la máxima distancia absoluta en una dimensión o en dos o n dimensiones.
- Puedes establecer umbrales para que recibas avisos cuando la deriva sea superior a lo que consideres aceptable.

Schema skew

Serving and training data don't conform to same schema:

- For example, `int != float`

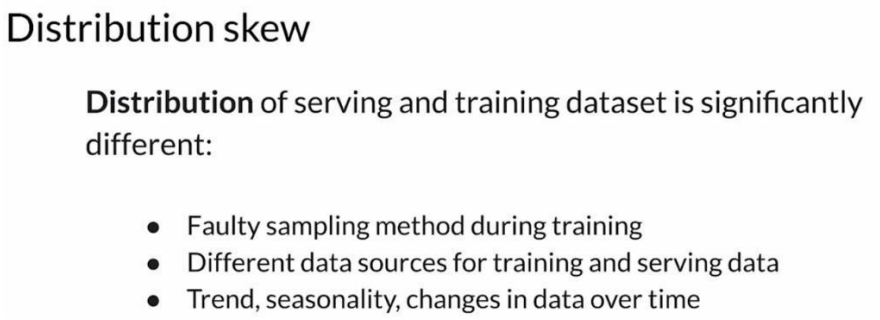
- La desviación del esquema se produce cuando los datos de servicio y de entrenamiento no se ajustan al mismo esquema.
- Por ejemplo, podría ser un cambio de tipo, un int, donde se espera un float, lo que podría ser un cambio en la propia función.

Feature skew

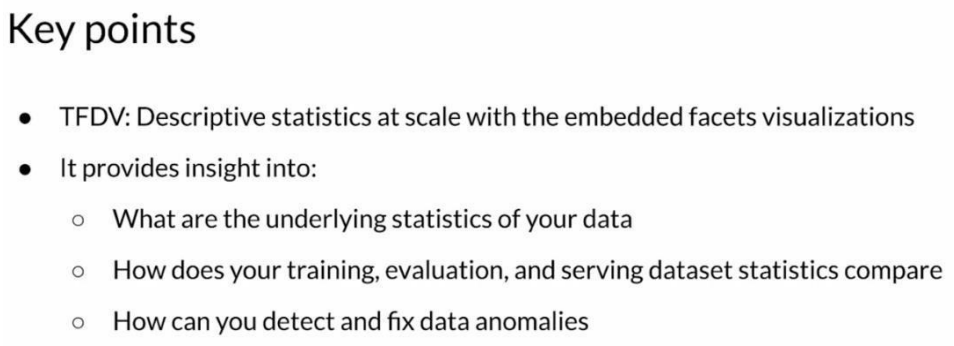
Training **feature values** are different than the serving **feature values:**

- Feature values are modified between training and serving time
- Transformation applied only in one of the two instances

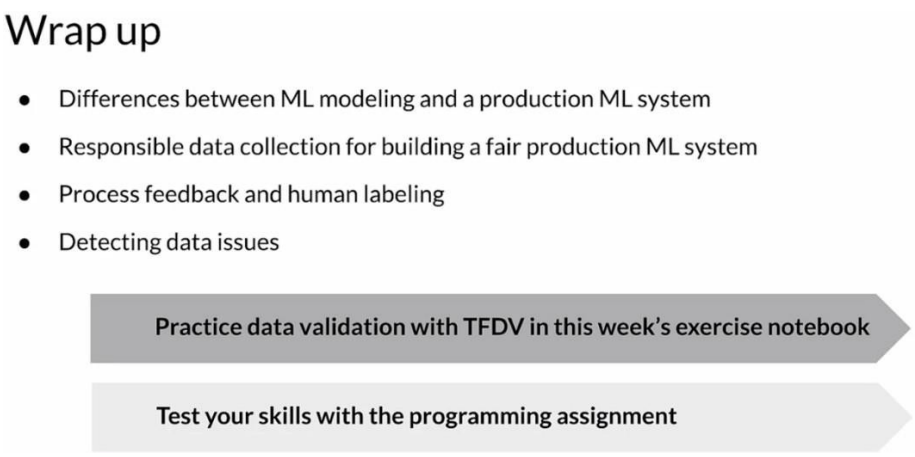
- El sesgo de las características son los cambios en los valores de las características entre el entrenamiento y el servicio.
- Puede ocurrir que el sistema utilice diferentes fuentes de datos durante el entrenamiento y el servicio, o que las cosas cambien también debido a la estacionalidad y la tendencia.
- A veces eso se debe simplemente a que tienes dos rutas de código diferentes cuando intentas hacer las mismas transformaciones, lo que hace que obtengas resultados diferentes como consecuencia.



- La inclinación de la distribución son los cambios en la distribución de las características individuales en el conjunto de datos.
- Las características que están en entrenamiento pueden tener un rango de 0 a 100 cuando lo estás entrenando, y luego en el momento de servir, estás viendo datos entre 5 y 600.
- Eso sería un cambio en la distribución de esa característica.
- Puede haber cosas como cambios en la media o en la mediana o en la desviación estándar, todos ellos son cambios en la distribución.
- Dependiendo de su gravedad, puede ser un problema o no. La pregunta es: ¿afecta al rendimiento de tu modelo lo suficiente como para que tengas que hacer cambios para intentar tenerlo en cuenta?



- TFDV le proporciona estadísticas descriptivas a escala. Recuerda que podríamos estar trabajando con petabytes de datos.
- También proporciona algunas visualizaciones para ayudarte a controlar y comprender realmente esos datos, para que entiendas las estadísticas subyacentes de tus datos y hagas comparaciones. ¿Cómo se comparan los conjuntos de datos de entrenamiento, evaluación y servicio?
- Sólo en términos de estadística, por ejemplo, ¿tienen la misma media? ¿Cómo se puede calcular y arreglar o detectar más bien y arreglar las anomalías de los datos?



Más información

- MLops
- Datos ciudadano de primera clase
- Aplicación para corredores
- Reglas del ML
- Sesgo en los conjuntos de datos
- Logstash
- Fluentd
- Registro de Google Cloud
- AWS ElasticSearch
- Monitor Azure
- TFDV
- Distancia de Chebyshev
- Konstantinos, Katsiapis, Karmarkar, A., Altay, A., Zaks, A., Polyzotis, N., ... Li, Z. (2020). Towards ML Engineering: Una breve historia de TensorFlow Extended (TFX). http://arxiv.o r g / a b s / 2 0 1 0 . 0 2 0 1 3
- Paleyes, A., Urma, R.-G., & Lawrence, N. D. (2020). Desafíos en el despliegue del aprendizaje automático: A survey of case studies. http://arxiv.org/abs/2011.09926
- Fracción de código ML: Sculley, D., Holt, G., Golovin, D., Davydov, E., & Phillips, T. (s.f.). Hidden technical debt in machine learning systems. Recuperado el 28 de abril de 2021, de Nips.cc https://papers.nips.cc/paper/2015/file/86df7dcfd896fcaf2674f757a2463eba-Paper.pdf