Chapter 2: Introduction to Machine Learning Systems Design

Chip Huyen

La Idea Principal: Diseñar un Sistema de ML es Mucho Más que Entrenar un Modelo

El capítulo 2 nos saca del mundo puramente algorítmico del *Machine Learning* (ML) y nos introduce en el mundo real de la **ingeniería de sistemas de ML**. La idea central es que construir un sistema de ML exitoso no se trata de obsesionarse con la precisión de un modelo, sino de adoptar un **enfoque de sistema holístico**.

Esto significa que debemos considerar todos los componentes juntos: el objetivo del negocio, los datos, la infraestructura, el monitoreo y, por supuesto, el modelo, para asegurarnos de que todo funcione en armonía para resolver un problema real.

---

1. Objetivos de Negocio y Objetivos de ML

Este es el punto de partida y, según la autora, el error más común en proyectos de ML que fracasan.

- **El Error Típico**: Los científicos de datos se emocionan mejorando las **métricas de ML** (como la exactitud de un 94% a un 94.2%).

- **La Dura Realidad**: A la mayoría de las empresas no les importa ese 0.2% de mejora. Solo les importan las **métricas de negocio**: ¿esa mejora se traduce en más ingresos, clientes más satisfechos o menores costos?

La lección clave es conectar ambos mundos. Antes de empezar, debes responder: ¿Qué métrica de negocio (ej: ingresos por publicidad, número de usuarios activos) se supone que este sistema de ML debe mover?

- **Ejemplo de Éxito**: Netflix no solo mide la precisión de su sistema de recomendación. Crearon una métrica puente llamada **"take-rate"** (tasa de aceptación): el número de veces que un usuario reproduce un título recomendado. Descubrieron

que un `take-rate` más alto se correlacionaba directamente con métricas de negocio importantes como más horas de visualización y menos cancelaciones de suscripción.

A veces, la única forma de saber si un modelo mejora el negocio es a través de la experimentación (como las pruebas A/B). Y hay que ser realistas: **el ML no es magia**. No transformará un negocio de la noche a la mañana. Google ha tardado décadas en perfeccionar sus sistemas. El retorno de la inversión depende mucho de la madurez de la empresa en la adopción de ML.

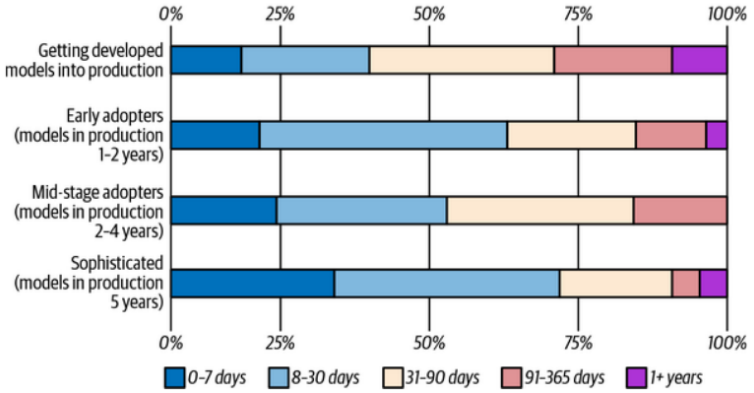

Figura 2-1. El tiempo que le toma a una empresa llevar un modelo a producción es proporcional al tiempo que lleva usando ML. Fuente: Adaptado de una imagen de Algorithmia.

---

2. Requisitos para los Sistemas de ML

Para construir un buen sistema, necesitamos definir sus requisitos. La autora se centra en cuatro características universales:

1. **Fiabilidad (Reliability)**: El sistema debe funcionar correctamente, incluso ante fallos (de hardware, software o errores humanos).

    - **El Peligro del "Fallo Silencioso"**: A diferencia del software tradicional que se "rompe" con un error 404, los sistemas de ML pueden fallar silenciosamente. Por ejemplo, Google Translate puede darte una traducción incorrecta de una frase, y si no conoces el idioma, nunca te darás cuenta de que está fallando.

2. **Escalabilidad (Scalability)**: El sistema debe ser capaz de manejar el crecimiento. Este crecimiento puede ocurrir de tres maneras:

- **Crecimiento en Tráfico**: Pasar de 10,000 a 10 millones de predicciones al día.

- **Crecimiento en Complejidad del Modelo**: Pasar de un modelo simple a una red neuronal gigante que necesita mucha más RAM.

- **Crecimiento en el Número de Modelos**: Pasar de tener un solo modelo para todos los usuarios a tener 8,000 modelos, uno para cada cliente empresarial. La gestión de un solo artefacto es muy diferente a la gestión de miles.

3. **Mantenibilidad (Maintainability)**: Muchas personas trabajarán en el sistema a lo largo del tiempo. Debe ser fácil de entender, modificar y mejorar.

   - **Claves**: Buen código documentado, versionado de todo (código, datos y artefactos), y modelos reproducibles para que otros puedan construir sobre tu trabajo.

4. **Adaptabilidad (Adaptability)**: El mundo cambia, y los datos también. El sistema debe poder evolucionar rápidamente para adaptarse a nuevos patrones de datos (*data distribution shifts*) y nuevos requisitos de negocio, idealmente sin interrumpir el servicio. Esto está muy ligado al concepto de *Continual Learning*.

---

3. El Proceso Iterativo

Construir un sistema de ML no es un proceso lineal (diseñar → construir → listo). Es un **ciclo iterativo y, a menudo, interminable**.

La autora cuenta su experiencia construyendo un modelo para predecir si se debe mostrar un anuncio. El proceso, que parecía simple, se convirtió en un ciclo de 13 pasos llenos de idas y venidas:

- Te das cuenta de que las etiquetas de los datos son incorrectas, así que vuelves a etiquetar.

- Te das cuenta de que el modelo está desactualizado, así que lo reentrenas con datos nuevos.

- Te das cuenta de que optimizaste la métrica equivocada (impresiones en lugar de clics), así que vuelves al paso 1.

Este ciclo caótico se puede simplificar en un proceso de 6 etapas interconectadas:

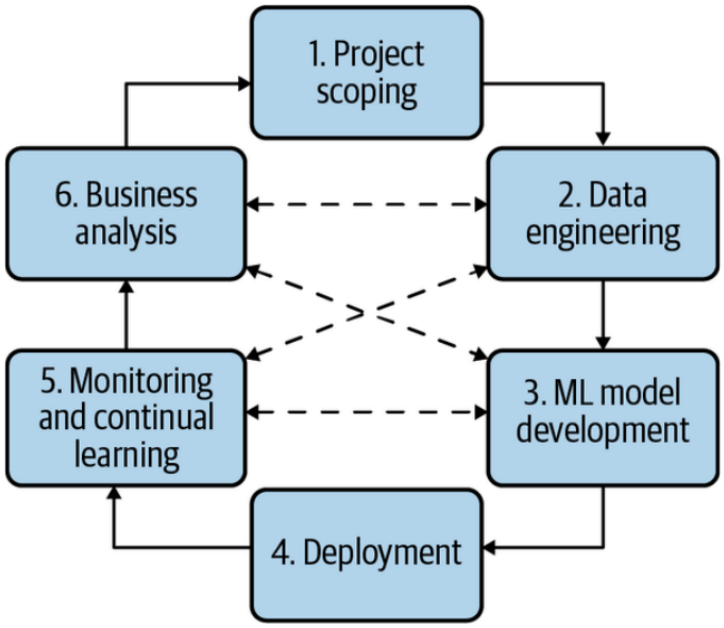

Figura 2-2. El proceso de desarrollo de un sistema de ML se parece más a un ciclo con muchas idas y venidas entre los pasos.

1. **Definición del Alcance del Proyecto (Project Scoping)**: Definir objetivos, restricciones y recursos.

2. **Ingeniería de Datos (Data Engineering)**: La base de todo. Recolectar, limpiar y etiquetar los datos.

3. **Desarrollo del Modelo de ML (ML Model Development)**: La parte que se enseña en los cursos: extraer características, seleccionar, entrenar y evaluar modelos.

4. **Despliegue (Deployment)**: Poner el modelo a disposición de los usuarios.

5. **Monitoreo y Aprendizaje Continuo (Monitoring and Continual Learning)**: Vigilar el rendimiento del modelo en producción y reentrenarlo cuando sea necesario.

6. **Análisis de Negocio (Business Analysis)**: Evaluar el rendimiento del modelo contra los objetivos de negocio. Los *insights* de este paso alimentan de nuevo la definición de nuevos proyectos, cerrando el ciclo.

---

4. Enmarcar los Problemas de ML (Framing)

Esta es una de las habilidades más importantes de un ingeniero de ML: traducir un problema de negocio vago en una tarea de ML bien definida.

- **Problema de Negocio**: "Nuestro servicio de atención al cliente es demasiado lento."
- **Esto NO es un problema de ML**. Un problema de ML se define por **entradas, salidas y una función objetivo**.
- **Enmarcando el Problema**: Tras investigar, descubres que el cuello de botella está en enrutar las peticiones al departamento correcto.
  - **Problema de ML**: Crear un modelo de **clasificación**.
  - **Entrada**: El texto de la petición del cliente.
  - **Salida**: El departamento al que debe ir (Contabilidad, Inventario, RRHH o TI).
  - **Función Objetivo**: Minimizar la diferencia entre el departamento predicho y el departamento real.

Tipos de Tareas de ML

El tipo de **salida** que necesitas define el tipo de tarea de ML.

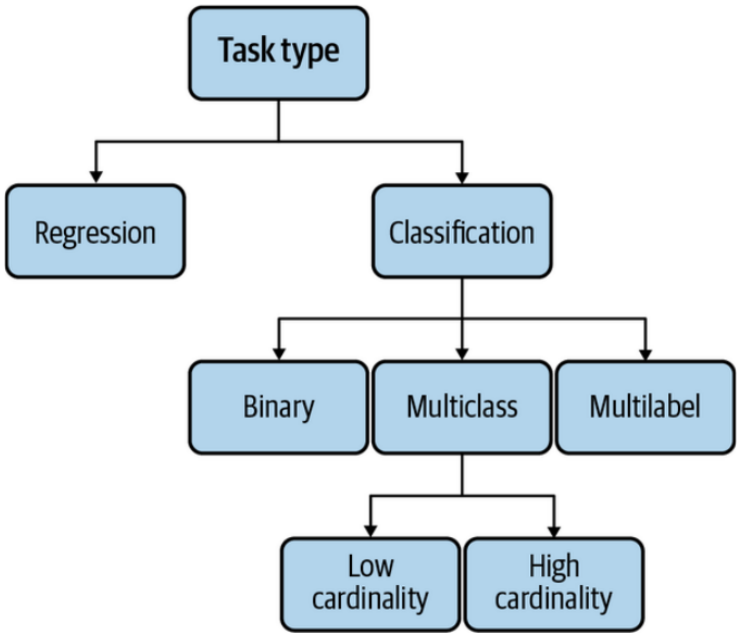

Figura 2-3. Tipos de tareas comunes en ML.

- **Regresión vs. Clasificación**:
  - **Regresión**: La salida es un valor numérico continuo (ej: el precio de una casa).
  - **Clasificación**: La salida es una categoría (ej: el correo es *spam* o *no spam*).
  - **Intercambiables**: Un problema puede enmarcarse de ambas maneras. La predicción de precios puede convertirse en una clasificación si agrupas los precios en rangos (ej: “100k-200k”, “200k-300k”).

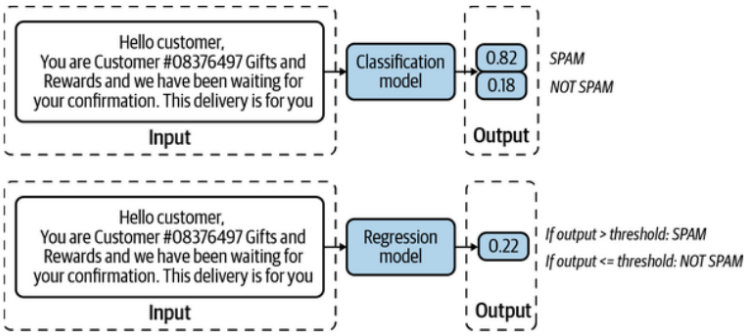

Figura 2-4. La tarea de clasificación de correo electrónico también se puede enmarcar como una tarea de regresión.

- **Subtipos de Clasificación**:
  - **Binaria**: Dos clases posibles (ej: *fraude* o *no fraude*).
  - **Multiclase**: Más de dos clases, pero una sola es correcta (ej: una imagen es un *perro*, un *gato* o un *pájaro*).
  - **Multietiqueta (Multilabel)**: Un ejemplo puede pertenecer a varias clases a la vez (ej: un artículo de noticias puede ser sobre *tecnología* Y *finanzas*).

Múltiples Maneras de Enmarcar un Problema

La forma en que enmarcas el problema tiene un impacto enorme en la dificultad y el mantenimiento del sistema.

- **Ejemplo**: Predecir la próxima app que un usuario abrirá.
  - **Enfoque Ingenuo (Clasificación)**: La entrada son las características del usuario, y la salida es una probabilidad para CADA app instalada en el teléfono. **Problema**: Cada vez que se instala una nueva app, ¡tienes que reentrenar todo el modelo!

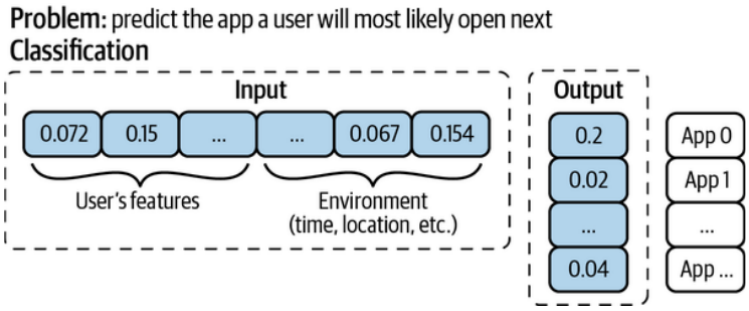

Figura 2-5. Dado el problema de predecir la próxima aplicación que un usuario abrirá, puedes enmarcarlo como un problema de clasificación. La entrada son las características del usuario y del entorno. La salida es una distribución sobre todas las aplicaciones del teléfono.

– **Mejor Enfoque (Regresión)**: La entrada son las características del usuario + las características de **una app específica**. La salida es un solo número (la probabilidad de que esa app sea abierta). **Ventaja**: Si se instala una nueva app, simplemente la pasas como una nueva entrada al modelo existente, sin necesidad de reentrenar.

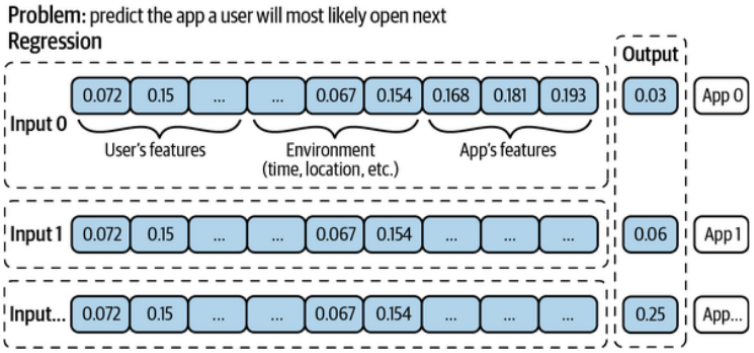

Figura 2-6. Dado el problema de predecir la próxima aplicación que un usuario abrirá, puedes enmarcarlo como un problema de regresión. La entrada son las características del usuario, del entorno y de una aplicación. La salida es un único valor entre 0 y 1 que denota la probabilidad de que el usuario abra la aplicación dado el contexto.

Funciones Objetivo

Un modelo de ML necesita una **función objetivo** (o *loss function*) para aprender. El objetivo es minimizar la pérdida causada por las predicciones incorrectas.

- **Múltiples Objetivos en Conflicto**: A veces, tienes objetivos contradictorios. Ejemplo: quieres maximizar el *engagement* en un feed de noticias, pero también minimizar la desinformación. Las publicaciones extremas suelen tener más *engagement*.

- **Solución (Desacoplar Objetivos)**: En lugar de entrenar un solo modelo complejo con una función de pérdida combinada, es mejor entrenar **dos modelos separados**:

  1. Un `modelo_de_calidad`.

  2. Un `modelo_de_engagement`.

  Luego, combinas sus puntuaciones con pesos ($\alpha * score\_calidad + \beta * score\_engagement$). La gran ventaja es que puedes ajustar los pesos α y β sin tener que reentrenar los modelos, lo que hace el mantenimiento mucho más fácil.

---

5. Mente vs. Datos (Mind Versus Data)

El capítulo termina con un debate filosófico fundamental en el ML: **¿Qué es más importante para el éxito, algoritmos más inteligentes (Mente) o cantidades masivas de datos (Datos)?**

- **Equipo "Mente sobre Datos" (Mind-over-Data)**: Liderado por figuras como Judea Pearl, argumentan que "los datos son profundamente tontos". Se necesitan diseños inteligentes y modelos causales para lograr un verdadero entendimiento.

- **Equipo "Datos sobre Mente" (Data-over-Mind)**: Liderado por figuras como Richard Sutton ("The Bitter Lesson") y Peter Norvig de Google. Su argumento es que, históricamente, los métodos generales que aprovechan la computación masiva y grandes volúmenes de datos siempre terminan superando a los enfoques que dependen del conocimiento humano y diseños intrincados. La famosa cita de Norvig sobre el éxito de la búsqueda de Google es: **"No tenemos mejores algoritmos. Simplemente tenemos más datos."**

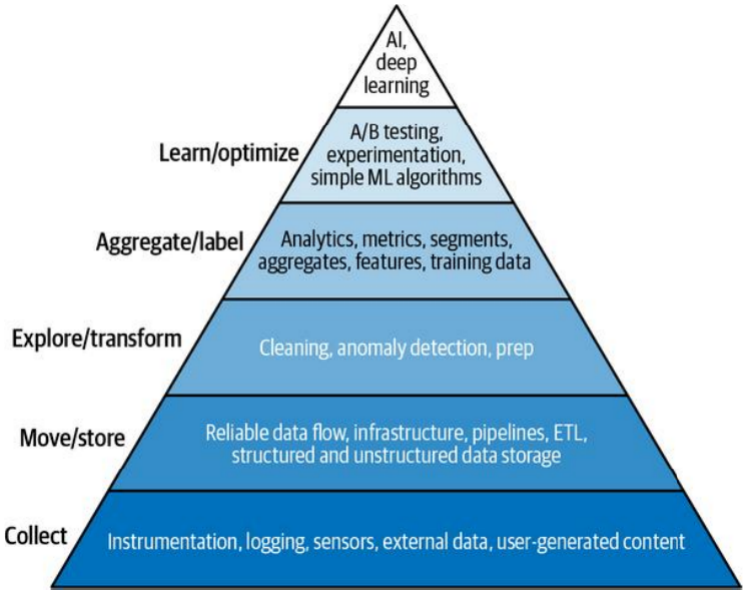

Figura 2-7. La jerarquía de necesidades de la ciencia de datos. Fuente: Adaptado de una imagen de Monica Rogati.

La pirámide de Monica Rogati (Figura 2-7) ilustra perfectamente este punto: la IA y el *deep learning* están en la cima, pero se sustentan sobre una base masiva de recolección, almacenamiento, limpieza y etiquetado de datos. Sin una base sólida, la cima se derrumba.

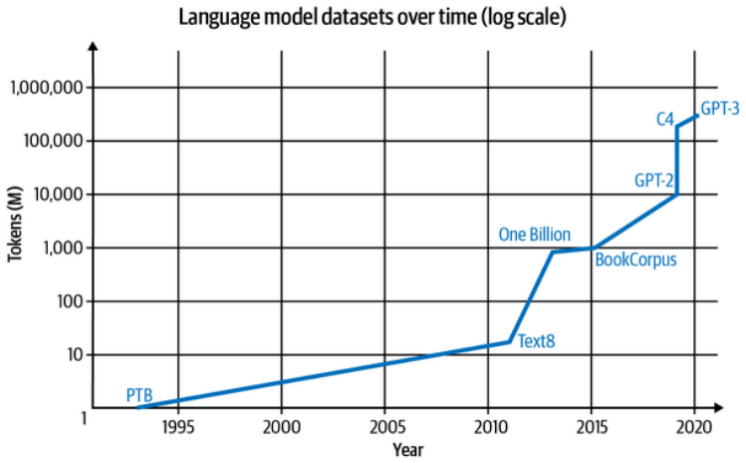

Figura 2-8. El tamaño de los conjuntos de datos (escala logarítmica) utilizados para modelos de lenguaje a lo largo del tiempo.

Conclusión de la autora: Independientemente de quién tenga la razón a largo plazo, hoy en día **los datos son esenciales**. La tendencia de la última década muestra que el éxito del ML depende cada vez más de la calidad y la cantidad de los datos. Sin embargo, más datos no siempre es mejor; datos de mala calidad pueden incluso dañar el rendimiento de un modelo.

Resumen Final del Capítulo

1. **Empezar por el Negocio**: Un proyecto de ML debe estar motivado por objetivos de negocio, que luego se traducen en objetivos de ML.

2. **Definir Requisitos**: Un buen sistema de ML debe ser fiable, escalable, mantenible y adaptable.

3. **Proceso Iterativo**: Construir un sistema de ML es un ciclo continuo, no una tarea de una sola vez.

4. **Enmarcar el Problema**: La habilidad de convertir un problema de negocio en una tarea de ML bien definida es crucial.

5. **El Papel de los Datos**: Aunque el debate "mente vs. datos" continúa, el progreso actual del ML depende fundamentalmente del acceso a grandes cantidades de datos de alta calidad.