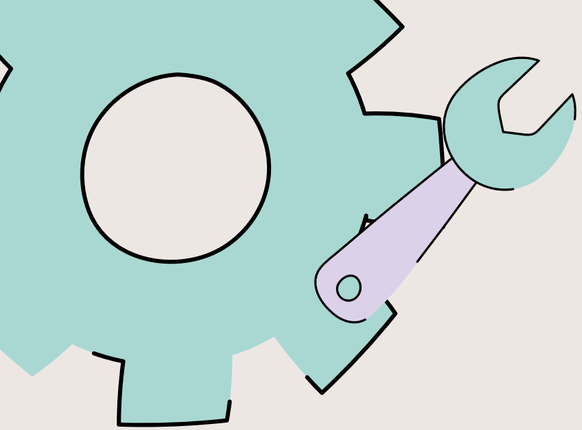
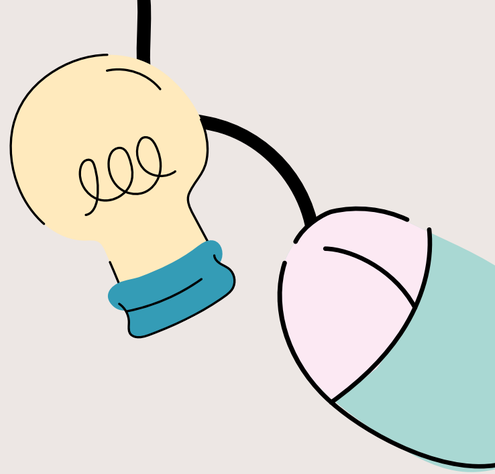
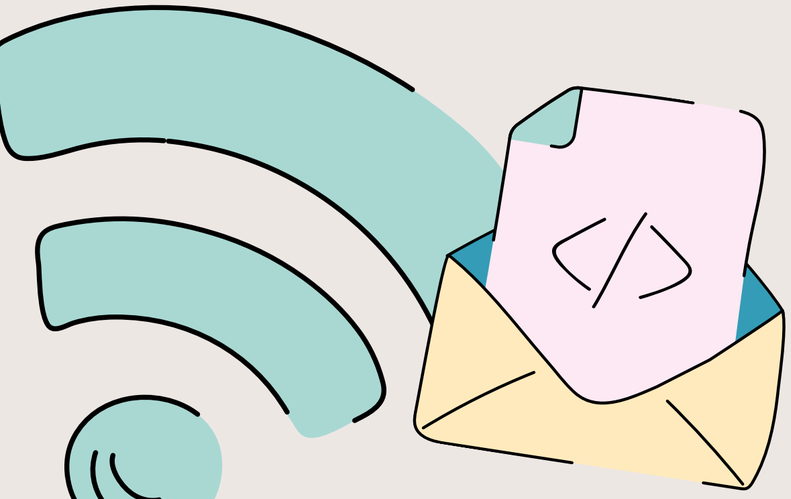
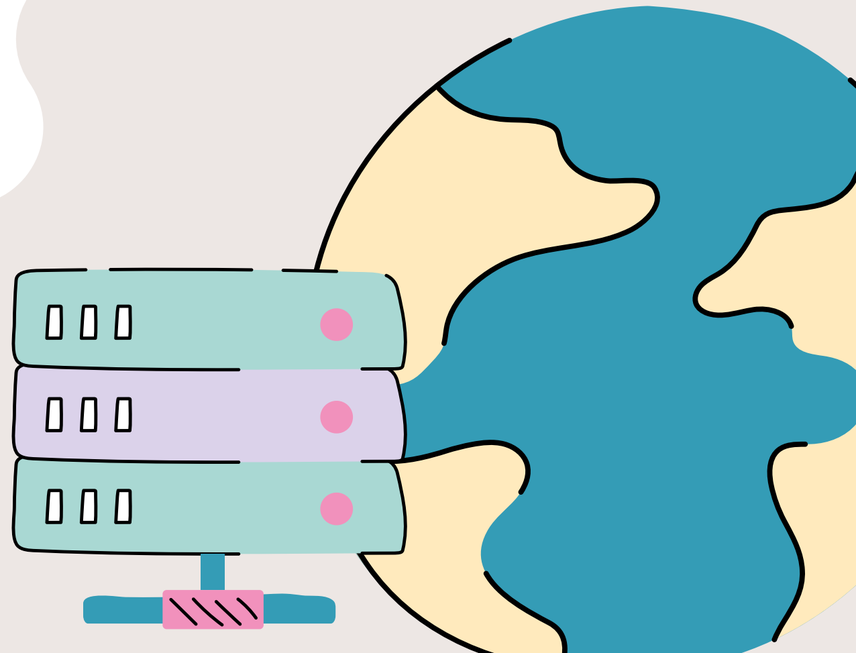




ROADMAP ML ENGINEER

De cero a implementar modelos en producción





Por Juan Duran

¿Qué hace realmente un ML Engineer?

Spoiler: no se pasa el día entero entrenando modelos.

Un **Machine Learning Engineer** está en la intersección entre el **desarrollo de software** y la **ciencia de datos**. Su trabajo no es solo crear modelos que funcionen, sino asegurarse de que esos **modelos** puedan ser integrados, escalados y mantenidos en **entornos reales**.

Algunas de sus tareas típicas:

- **Colaborar** con científicos de datos y product managers para entender el problema.
- **Desarrollar** pipelines robustos de datos.
- **Elegir, entrenar y ajustar** modelos de ML.
- **Hacer testing** y validación de modelos.
- **Desplegar** y **monitorizar** modelos en producción.

Fundamentos

Antes de pensar en redes neuronales o modelos en la nube, necesitas una base sólida en tres áreas:

- **Programación (Python sí o sí)**

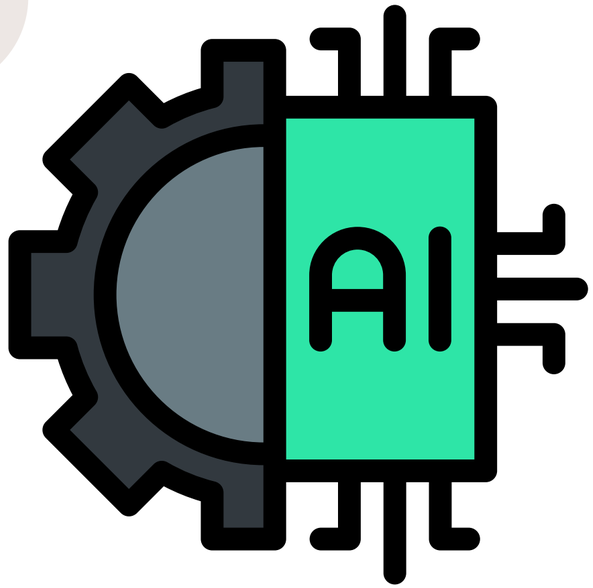
Aprende estructuras, funciones, clases, testing y buenas prácticas. Sin eso, todo lo demás cojea.

- **Matemáticas para ML**

No necesitas ser un matemático, pero sí entender álgebra lineal, cálculo básico y sobre todo estadística. Te va a ayudar muchísimo cuando entrenes modelos.

- **Pensamiento computacional**

Saber cómo descomponer problemas, pensar en eficiencia, entender algoritmos y estructuras de datos. Esto te hará mejor ingeniero, no solo mejor en ML.



Manipulación y preparación de datos

Antes de entrenar cualquier modelo, vas a pasar una cantidad sorprendente de tiempo **limpiando** y **procesando datos**. Es el trabajo sucio, pero esencial.

Aprende a:

- Usar **pandas**, **NumPy** y **SQL** para explorar datos.
- Identificar **outliers**, **nulos**, **duplicados**, datos mal formateados.
- Aplicar **transformaciones** como escalado, codificación, normalización.
- Construir **pipelines** reproducibles con scikit-learn, Feature-engine o sklearn-pandas.

Los modelos buenos nacen de buenos datos.

Machine Learning

Aquí empieza la **magia** (y también los errores si no tienes claro el paso anterior).

En esta etapa deberías:

- **Conocer los modelos más usados**: regresión lineal, regresión logística, árboles de decisión, random forest, SVM, k-NN.
- Entender **cuándo usar** cada uno y por qué.
- Medir el **rendimiento** de los **modelos**: accuracy, recall, precision, F1, ROC.
- **Validar** con técnicas como cross-validation, grid search y hyperparameter tuning.

La clave no es usar el modelo más complejo, sino el más adecuado.

Deep Learning

El **deep learning** no es obligatorio, pero es una **herramienta** muy **potente** si tienes muchos datos o problemas específicos como imágenes, texto o series temporales.

Temas que deberías cubrir:

- Redes neuronales básicas con Keras o PyTorch.
- CNNs para imágenes.
- RNNs, LSTM y Transformers para secuencias y texto.
- Preentrenamiento y fine-tuning de modelos.

Importante: si no dominas ML clásico, el deep learning te va a parecer magia negra. Paso a paso.

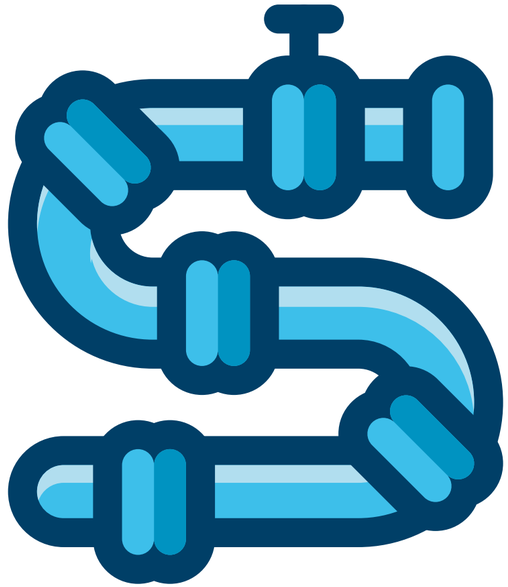
Ingeniería de ML: producción, testing, pipelines

Aquí es donde realmente te diferencias de un Data Scientist.

Un ML Engineer:

- **Automatiza pipelines** de entrenamiento y predicción (con herramientas como MLflow, Airflow o Prefect).
- **Versiona datos, modelos y experimentos**.
- Hace **testing** de modelos.
- Usa buenas prácticas de desarrollo: **CI/CD, Docker, Git**, etc.
- **Despliega modelos** (API REST, FastAPI, Flask, Lambda).

Aprender a poner un modelo en producción es lo que realmente te convierte en ingeniero.



Herramientas y frameworks

A medida que avanzas, hay herramientas que se vuelven indispensables:

- **scikit-learn** – Tu caja de herramientas de ML.
- **TensorFlow / PyTorch** – Para Deep Learning.
- **MLflow / DVC** – Para tracking de experimentos.
- **Docker + FastAPI** – Para crear servicios con modelos.
- **Cloud (GCP, AWS, Azure)** – Para escalar en serio.
- **Airflow / Prefect** – Para orquestar procesos de datos.

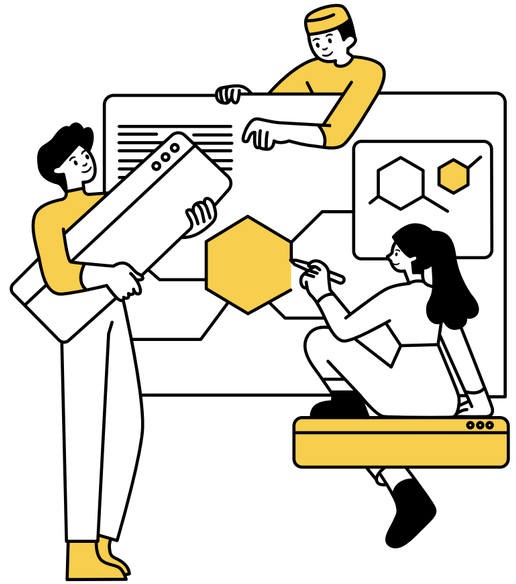
No necesitas todas desde el principio. Pero tenlas en el radar.

Proyectos que puedes (y deberías) hacer

No hay mejor forma de aprender que ensuciándote las manos con **proyectos**. Aquí algunas ideas:

- **Predicción** de churn de clientes con ML clásico.
- **Clasificador** de imágenes de flores con CNNs.
- **Recomendador** de películas usando embeddings.
- Modelo de **series temporales** para ventas.
- **Proyecto end-to-end con ML + API + Docker**.

Elige problemas que te interesen y trata de llevarlos a producción (aunque sea en tu máquina local).



Qué suelen valorar las empresas...

Además del conocimiento técnico, las empresas buscan:

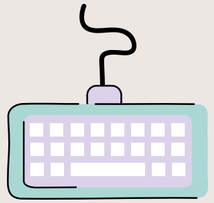
- Capacidad para **comunicar** resultados.
- **Enfoque** en **producción**, no en notebooks eternos.
- **Pensamiento crítico** sobre datos y modelos.
- Conocimiento de **buenas prácticas de software**.
- **Experiencia con proyectos reales** (aunque sean personales).
- Curiosidad por **aprender** y **mejorar** constantemente.

Mucho de esto no se aprende en cursos, sino en la práctica.

Consejos finales si estás empezando

- **No te frustres** si no entiendes todo al principio. Es normal.
- **Aprende poco a poco** y consolida lo básico antes de pasar a lo avanzado.
- **Haz proyectos** aunque no te sientas “listo”. Justo ahí es donde más aprendes.
- **Comparte** tu proceso. Escribir sobre lo que aprendes refuerza tu conocimiento y te da visibilidad.
- **Busca comunidad**: foros, Discord, LinkedIn, lo que te funcione.

Esto es un maratón, no un sprint. Disfruta el proceso.



Gracias

Por Juan Duran

“Coding, Gaming and Leveling Up”