CAPÍTULO I

INTRODUCCIÓN A LA CIENCIA DE DATOS E INGENIERÍA DE DATOS

En la última década hemos sido testigos de un crecimiento vertiginoso en el campo del aprendizaje automático. Los modelos han aprendido a reconocer rostros, traducir idiomas, detectar fraudes, anticipar comportamientos y hasta imitar la voz humana con una precisión que, hace unos años, parecía impensable. Sin embargo, detrás de esa aparente perfección tecnológica se esconde una verdad que muchos investigadores, entre ellos *Andrew Ng*, han resaltado con claridad: el verdadero éxito de un modelo no depende únicamente de su arquitectura o de cuántas capas tenga, sino de la calidad de los datos con los que se alimenta.

Y es que los datos son mucho más que números o registros: son historias, contextos y decisiones humanas convertidas en información. Cuando esos datos son incompletos, sesgados o mal comprendidos, los modelos aprenden una versión distorsionada de la realidad. Por eso, en los últimos años, la comunidad científica ha comenzado a mirar más allá del modelo y a preguntarse por el corazón mismo de todo sistema inteligente: los datos.

De esa reflexión surge la idea de que no basta con diseñar algoritmos cada vez más potentes; es necesario comprender el origen de los datos, su estructura, su limpieza y su significado. Esa es, precisamente, la tarea del ingeniero de datos: construir los caminos por los que la información viaja, cuidando que lleguen limpias, ordenadas y listas para convertirse en conocimiento.

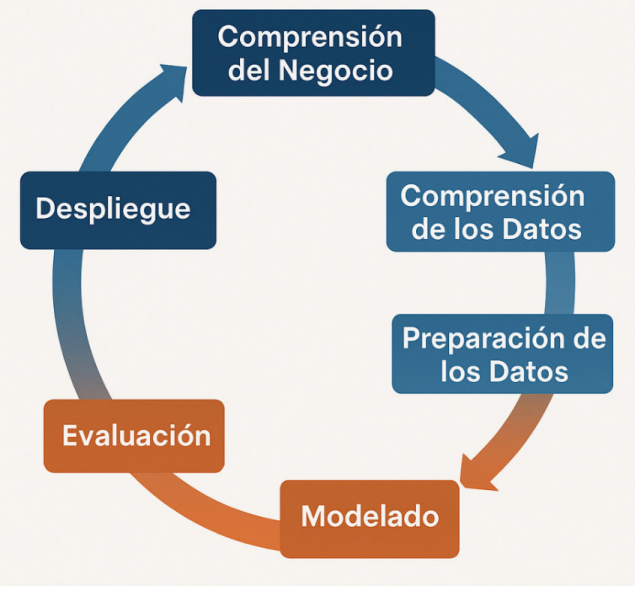
Este capítulo busca invitarte a mirar la ciencia de datos con ojos nuevos. A no verla solo como un conjunto de técnicas, sino como una manera de pensar el

mundo a través de la información. Aquí hablaremos del ciclo de vida del dato, del papel del ingeniero que lo acompaña desde su captura hasta su transformación, y de los dos grandes paradigmas que hoy guían la inteligencia artificial: el *model-centric* y el *data-centric*. Más que definiciones, encontrarás aquí una conversación: una oportunidad para entender por qué los datos importan tanto como, o incluso más que, los modelos que intentan aprender de ellos.

1.1. Fundamentos y ciclo de vida de los datos

Una de las metodologías más utilizadas por los científicos y los ingenieros de datos se denomina CRISP-DM (*Cross Industry Standard Process for Data Mining*). Fue desarrollada hace más de veinte años, pero sigue siendo plenamente vigente como marco de trabajo para proyectos de ciencia de datos. Más que un conjunto de pasos rígidos, CRISP-DM propone un ciclo flexible y reflexivo: comprender el problema, conocer los datos, prepararlos, modelar, evaluar y desplegar. Cada fase representa un momento en la vida del dato, una etapa donde la ingeniería, la estadística y la interpretación se encuentran. Estas fases se presentan en la Figura 1.

Figura 1. Ciclo de vida de los datos: Metodología CRISP-DM. Generada con ChatGPT por el autor.



Todo comienza con la **comprensión del problema o del contexto**. Antes de escribir una sola línea de código, el equipo debe entender cuál es el propósito de los datos, es decir, qué se pretende predecir u obtener con su uso. Esta fase da sentido al proyecto: define el objetivo y permite que las siguientes decisiones técnicas mantengan coherencia con el propósito inicial.

Posteriormente llega la **comprensión de los datos** (*data understanding*). Es el momento de realizar un **EDA (Exploratory Data Analysis)**, que consiste en conocer su comportamiento estadístico, su relación con otras características (*features*) y con la salida (si existe). Además, en esta etapa se realizan tareas esenciales como la limpieza de datos, que permite detectar valores faltantes, eliminar duplicados y corregir errores que podrían afectar el modelado posterior.

A continuación, se desarrolla la fase de **preparación de los datos**, en donde la "magia" de los ingenieros y científicos de datos cobra vida. Se corrigen errores, se unifican formatos y se eliminan inconsistencias, pero también se mejoran y construyen nuevas variables mediante el ***feature Engineering*** **(FE)**, un proceso que busca crear o seleccionar las características más representativas para que el modelo aprenda de forma más efectiva. En esencia, esta etapa convierte los datos crudos en información útil, lista para el análisis, asegurando que lo que llegue al modelo refleje con fidelidad la realidad que se intenta comprender.

Con los datos listos, se pasa a la fase de **modelado**, donde entran en juego las técnicas de aprendizaje automático y aprendizaje profundo. Se prueban distintos algoritmos, se ajustan hiperparámetros y se comparan resultados. Pero aquí aparece una verdad importante: el modelado no es el final del proceso, sino una consecuencia de las decisiones tomadas en las fases anteriores. Un modelo solo puede aprender aquello que los datos le permiten aprender.

Después llega la **evaluación**, un momento de análisis crítico. No basta con obtener una buena métrica; es necesario revisar si el modelo responde a la pregunta original, si los datos fueron suficientes o si existen sesgos que puedan distorsionar los resultados. Esta etapa tiene un carácter científico y ético: invita a detenerse, revisar y validar antes de avanzar.

Finalmente, el ciclo culmina con la **implementación o despliegue**, cuando los resultados del proyecto se integran en un entorno real o se presentan

como parte de una investigación aplicada. Pero el ciclo no termina allí. CRISP-DM propone un retorno constante a las fases anteriores, una mejora continua que reconoce que los datos, como la realidad que representan, están siempre en movimiento.

1.2. Ingeniero de datos vs. Científico de datos vs. Analista de datos

En el ecosistema contemporáneo de la ciencia de datos, tres perfiles articulan el ciclo completo de generación de conocimiento: el ingeniero de datos, el científico de datos y el analista de datos. Aunque sus funciones a veces se superponen, cada uno representa una mirada distinta sobre la misma materia prima: el dato. Comprender esas diferencias permite construir equipos más sólidos y sistemas más confiables.

El **ingeniero de datos** diseña y mantiene la infraestructura que permite que los datos fluyan. Se encarga de construir pipelines, integrar fuentes, garantizar la calidad y disponibilidad de la información.

El **científico de datos** transforma esos datos en conocimiento. Explora, limpia, crea *features*, entrena modelos y evalúa resultados.

El **analista de datos**, finalmente, convierte ese conocimiento en comprensión y acción. Interpreta resultados, comunica hallazgos y los traduce en decisiones.

En la Tabla 1 se presenta una comparación entre los tres roles descritos anteriormente.

Tabla 1. Tabla comparativa entre Ingeniero de datos, Científico de Datos y Analista de datos.

Criterio	Ingeniero de Datos	Científico de Datos	Analista de Datos
Enfoque principal	Infraestructura y flujo del dato	Modelado, experimentación y descubrimiento de patrones	Interpretación y comunicación de resultados
Responsabilidades clave	Integración de fuentes, construcción de pipelines, aseguramiento de calidad, almacenamiento y orquestación	Limpieza analítica, *EDA*, *feature engineering*, entrenamiento y validación de modelos, evaluación de métricas	Análisis descriptivo, consultas, visualización, elaboración de reportes y *dashboards*
Competencias técnicas	SQL, Python, Spark, Kafka, Airflow, DVC, bases de datos, arquitecturas distribuidas	Python, R, pandas, scikit-learn, TensorFlow, PyTorch, estadística, machine learning	SQL, Excel, Power BI, Tableau, herramientas de visualización y *storytelling*
Tipo de limpieza de datos	Limpieza estructural (formatos, duplicados, integridad)	Limpieza analítica (*outliers*, sesgos, coherencia estadística)	Validación de coherencia y consistencia visual
Participación en el EDA	Soporte en la estructuración de datos para análisis	Ejecución principal: análisis exploratorio, correlaciones, visualización y generación de hipótesis	Interpretación de resultados exploratorios
Participación en FE	Implementa transformaciones recurrentes o automatizadas en pipelines	Diseña, crea y evalúa las características más relevantes para el modelo	Apoya en la interpretación y documentación de *features*

Criterio	Ingeniero de Datos	Científico de Datos	Analista de Datos
Herramientas y entornos	Airflow, Prefect, Kafka, DVC, bases SQL/NoSQL, sistemas cloud	Jupyter, Python, R, frameworks de ML, Git, MLflow	Power BI, Tableau, Google Data Studio, hojas de cálculo
Objetivo final	Garantizar datos confiables, accesibles y reproducibles	Extraer conocimiento útil y desarrollar modelos predictivos o explicativos	Traducir resultados técnicos en información clara para la toma de decisiones

1.2.1. **Límites y sinergias entre roles**

En la práctica, las fronteras entre estos tres perfiles son débiles. En entornos de investigación o *startups*, un mismo profesional puede desempeñar varios de estos papeles. Sin embargo, la distinción conceptual sigue siendo útil:

- El ingeniero asegura la existencia y estructura del dato.
- El científico le da sentido y valor predictivo.
- El analista le otorga contexto y aplicabilidad.

Estas tres perspectivas conforman una secuencia que une la ingeniería, la ciencia y la interpretación. Cuando los límites se entienden como zonas de colaboración, no de división, el flujo del dato se convierte en un proceso integral de conocimiento.

1.2.2. **Sobre la transformación y el FE**

A menudo se confunde la *T* del proceso ETL (Extract, Transform, Load) con el Feature Engineering (FE)**.**

Aunque ambos implican transformar datos, su propósito es distinto:

- En ETL, la transformación busca usabilidad operacional**:** estandarizar, limpiar o estructurar datos para su almacenamiento o consulta.

- En Feature Engineering, la transformación busca utilidad analítica o predictiva: crear variables que representen mejor los fenómenos o mejoren el aprendizaje del modelo.

Por tanto, el *feature engineering* puede verse como una extensión inteligente de la ingeniería de datos, donde la técnica se combina con la intuición científica. Su ubicación natural está en la frontera entre ambos mundos: depende de la infraestructura creada por el ingeniero, pero responde a la curiosidad analítica del científico.

1.2.3. **Aplicación práctica en investigaciones del autor**

La visión que articula este libro no surge únicamente del estudio teórico del ciclo de vida de los datos, sino también de una trayectoria investigativa que ha integrado, en la práctica, los tres grandes roles del ecosistema analítico: el ingeniero, el científico y el analista de datos. Los proyectos desarrollados en los últimos años, constituyen ejemplos concretos de cómo la ingeniería de datos se expande hacia lo científico y lo interpretativo, formando un continuo de conocimiento aplicado.

En el contexto investigativo, la ingeniería de datos abarca no solo la integración y transformación de información, sino también la construcción deliberada de datasets. Esta tarea implica diseñar colecciones de datos representativas, documentadas y reproducibles, donde cada decisión, desde la selección de fuentes hasta el formato de almacenamiento, constituye un acto de ingeniería. En proyectos como *h-voice* (*A dataset of histograms of original and fake voice recordings*) y *CG-1050* (*A dataset of 1050-tampered color and grayscale images*), la creación de los conjuntos de datos se concibe como un proceso estructurado de diseño y validación que traduce la teoría en evidencia experimental tangible.

En el ámbito de la ciencia de datos, el primer trabajo representativo fue *Deep4SNet*, donde se desarrolló un proceso de *feature engineering* dual: se extrajeron valores estadísticos de los audios (media, varianza, asimetría y curtosis) para alimentar un modelo de *machine learning* tradicional, y se generaron histogramas de las grabaciones para su procesamiento mediante una red neuronal convolucional (*CNN*). Este estudio marcó una transición entre la

caracterización clásica y la representación profunda del audio, evidenciando cómo la forma del dato define la capacidad del modelo para aprender. Posteriormente, *FakeVoiceFinder* consolidó esta línea de investigación al integrar múltiples representaciones espectrales (Mel, log, DWT y CQT) dentro de un marco abierto y reproducible, disponible en https://github.com/DEEP-CGPS/FakeVoiceFinder/tree/main. Este *framework* permitió comparar arquitecturas, medir el impacto de las transformaciones y establecer una base metodológica sólida para la detección de audio sintético, convirtiéndose en un ejemplo de ingeniería de datos aplicada a la investigación científica y a la generación de conocimiento reproducible.

En el nivel analítico, el trabajo *Is My Pruned Model Trustworthy? PE-Score: A New CAM-Based Evaluation Metric* representa una forma avanzada de análisis e interpretación de resultados. Este estudio propuso una métrica visual y cuantitativa basada en *Class Activation Maps* (CAM) para evaluar la coherencia y confiabilidad de los modelos podados, integrando interpretabilidad y rendimiento en un mismo marco evaluativo. De este modo, la labor del analista de datos trasciende la visualización descriptiva y se convierte en una evaluación crítica del comportamiento interno de los modelos, aportando transparencia y explicabilidad al proceso científico.

En conjunto, estas líneas de investigación demuestran que los roles de ingeniero, científico y analista de datos no son compartimentos estancos, sino dimensiones interdependientes de una misma práctica de conocimiento. En el contexto académico, esta convergencia permite que la ingeniería de datos trascienda su definición operacional y se convierta en un método de indagación científica: una manera estructurada, medible y ética de descubrir patrones, validar hipótesis y construir sentido a partir de la información.

1.3. **Machine Learning: problemas, métricas y validación**

El aprendizaje automático (*Machine Learning*) constituye una de las áreas más dinámicas de la ingeniería de datos, orientada a construir modelos capaces de aprender patrones a partir de la información disponible. Su aplicación no se limita a la selección de un algoritmo, sino que comienza con la identificación del tipo de problema que se desea resolver. En términos generales, los modelos de *Machine Learning* pueden agruparse en tres grandes categorías: regresión**,**

clasificación **y** clustering, cada una con un propósito y una naturaleza de salida diferentes.

En los problemas de **regresión**, el objetivo consiste en predecir valores numéricos continuos a partir de variables de entrada. Estos modelos estiman relaciones funcionales entre las características y la salida, buscando minimizar la diferencia entre el valor real y el predicho. Ejemplos clásicos incluyen la estimación del precio de un bien, la predicción de temperatura o la modelación de una señal en el tiempo. Su salida es, por tanto, un número real que representa una magnitud o tendencia.

En los problemas de **clasificación**, el propósito es asignar a qué clase o categoría pertenece el dato de entrada. El modelo aprende de los *features* de los datos y produce una salida discreta, la cual puede ser binaria o multiclase. En el caso binario tenemos como ejemplos *spam/no spam*, *voz real/voz sintética* o *tumor benigno/maligno*. En casos multiclase, puede predecir a qué animal pertenece una foto, de un grupo finito de opciones.

Por su parte, los problemas de **clustering** pertenecen al ámbito del *aprendizaje no supervisado*, donde el modelo no dispone de etiquetas previas (es decir, los datos no se entregaron con valores de salida tipo regresión o clase) y su tarea consiste en descubrir patrones o agrupaciones naturales dentro de los datos que permita agruparlos. El resultado no es una predicción explícita, sino la asignación de pertenencia a grupos o clústeres que comparten similitudes estructurales. Esta técnica se utiliza, por ejemplo, en la segmentación de clientes, la agrupación de señales o la detección de comportamientos anómalos.

Cada uno de estos tipos de problemas determina no solo el enfoque algorítmico, sino también las métricas de evaluación, los criterios de validación y la interpretación de resultados**.** En la práctica, comprender qué se espera del modelo (ej. un valor continuo, una categoría previamente definida o una estructura oculta) es el primer paso para construir experimentos de *Machine Learning* coherentes, reproducibles y científicamente sólidos.

1.3.1. Métricas de clasificación y regresión

Como se mencionó previamente, el tipo de problema de *Machine Learning* determina las métricas adecuadas para evaluar el rendimiento del modelo. En problemas de regresión, las métricas más comunes son:

- MAE (Mean Absolute Error)
- MSE (Mean Squared Error)
- RMSE (Root Mean Squared Error)
- R^2 (coeficiente de determinación)

Estas permiten cuantificar el error entre los valores reales y los valores predichos.

En problemas de clasificación, las métricas derivan de la matriz de confusión, la cual registra la cantidad de aciertos y errores por clase. Para el caso binario, la matriz se representa así:

Cuando se tiene clasificación binaria, la matriz de confusión es la siguiente:

		Predice 1	Predice 0
Real	1	TP	FN
	0	FP	TN

Donde

- TP (*True Positives*): instancias de la clase 1 correctamente clasificadas.
- TN (*True Negatives*): instancias de la clase 0 correctamente clasificadas.
- FP (*False Positives*): muestras de la clase 0 clasificadas incorrectamente como 1.
- FN (False Negatives): muestras de la clase 1 clasificadas incorrectamente como 0.

A partir de estos valores, se definen las siguientes métricas:

$$accuracy\ (acc) = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision\ (P) = \frac{TP}{TP + FP}$$

$$Recall\ (R) = \frac{TP}{TP + FN}$$

Es importante notar que *Precision* y *Recall* comparten el mismo numerador (TP), diferenciándose únicamente por el denominador, el cual representa la naturaleza del error que cada métrica penaliza.

Para P, es:

		Predice 1	Predice 0
Real	1	TP	FN
	0	FP	TN

Y para R, es:

		Predice 1	Predice 0
Real	1	TP	FN
	0	FP	TN

De acuerdo con lo anterior:

- *Accuracy* (precisión global): proporción de predicciones correctas sobre el total de muestras.
- *Precision*: fracción de predicciones positivas que realmente son positivas.
- *Recall* (sensibilidad): fracción de positivos reales que el modelo logra identificar.

Adicionalmente, se puede calcular la media armónica entre *P* y *R*, conocida como *F1-score*, así:

$$F1_{score} = 2\,\frac{P * R}{P + R}$$

1.3.2. Métricas para clasificación multiclase

En problemas de clasificación multiclase (con K clases), la matriz de confusión se extiende a una matriz de tamaño K × K, donde cada fila representa la clase real y cada columna la clase predicha. A diferencia del caso binario, los valores TP, FP y FN deben calcularse por clase, aplicando un enfoque uno contra todos (one-vs-all).

Cálculo de métricas por clase (para cada clase *i*):

- TPi: número de muestras de la clase i correctamente clasificadas.
- FNi: muestras de la clase i clasificadas como otra clase.
- FPi: muestras de otras clases clasificadas como clase i.
- TNi: combinaciones correctas restantes.

Las métricas por clase se definen como:

$$P_i = TPi\,/\,(TPi + FPi)$$

$$R_i = TPi\,/\,(TPi + FNi)$$

$$F1_i = 2 * (P_i * R_i)\,/\,(P_i + R_i)$$

Dado que en multiclase se tienen métricas por clase, se requiere un método para obtener un valor agregado global. Los más utilizados son:

a) *Macro-average*
Promedio simple de las métricas por clase.

$$P\_macro = \frac{1}{K}\sum_{i=1}^{K} P_i$$

$$R\_macro = \frac{1}{K}\sum_{i=1}^{K} R_i$$

$$F1\_macro = \frac{1}{K}\sum_{i=1}^{K} F1_i$$

La ventaja de *macro-average* es que da el mismo peso a todas las clases, lo cual es útil cuando todas las clases están balanceadas. Por otra parte, la desventaja es que puede ser engañoso cuando hay clases muy pequeñas, es decir, cuando el problema es desbalanceado.

b) *Micro-average*

En problemas de clasificación con múltiples clases, y especialmente cuando existe desbalance, las métricas *micro-average* permiten evaluar el desempeño global del modelo acumulando los verdaderos positivos, falsos positivos y falsos negativos de todas las clases antes de calcular cada métrica. Esto es especialmente útil cuando el dataset presenta desbalance entre clases, ya que otorga a cada ejemplo la misma importancia y evita que las clases mayoritarias dominen el cálculo.

$$P\_micro = (\Sigma TPi) / (\Sigma TPi + \Sigma FPi)$$

$$R\_micro = (\Sigma TTPi) / (\Sigma TPi + \Sigma FNi)$$

$$F1\_micro = 2 * (P\_micro * R\_micro) / (P\_micro + R\_micro)$$

c) *Weighted-average*

El enfoque *weighted-average* calcula las métricas ponderando el desempeño de cada clase según su tamaño relativo dentro del dataset. A diferencia del *macro-average*, que asigna el mismo peso a todas las clases, el *weighted-average* refleja la proporción real de cada categoría. Esto evita que las métricas se distorsionen cuando existen clases muy pequeñas o altamente desbalanceadas.

En conjunto, las métricas *weighted-average* ofrecen una representación más fiel del rendimiento global del modelo cuando el dataset presenta distribuciones desiguales entre clases.

$$P\_weighted = \Sigma\left((N_i / N_{total}) \cdot P_i\right)$$

$$R\_weighted = \Sigma\left(((N_i / N_{total}) \cdot R_i\right)$$

$$F1\_weighted = \Sigma\left((N_i / N_{total}) \cdot F1_i\right)$$

1.3.3. Importancia de las métricas en la interpretación de resultados

Con el siguiente ejemplo ilustraremos la necesidad de seleccionar adecuadamente las métricas al momento de interpretar qué tan bien lo está haciendo un modelo. Partamos del siguiente caso:

Se tiene un problema de clasificación binaria de detección de cáncer de mama. El 90 % de las muestras no tienen cáncer y el 10 % restante sí. Asumimos que la clase 1 corresponde a “cáncer” y que la clase 0 corresponde a “saludable”. El modelo de clasificación arrojó la siguiente matriz de confusión:

		Predice 1	Predice 0
Real	1	2	8
	0	2	88

Si evaluamos el desempeño del clasificador con *acc*, tendríamos que:

$$acc = \frac{2 + 88}{2 + 8 + 2 + 88} = \frac{90}{100} = 0.9$$

Es decir, clasifica correctamente el 90 % de los casos. No obstante, que el modelo se equivoque en la mayoría de los casos de cáncer no es deseable.

Vamos entonces a revisar las métricas P y R:

$$P = \frac{2}{2+2} = \frac{2}{4} = 0.5 P = \frac{2}{2+2} = \frac{2}{4} = 0.5, \quad \text{y} \quad R = \frac{2}{2+8} = \frac{2}{10} = 0.2$$

$$R = \frac{2}{2+8} = \frac{2}{10} = 0.2$$

Es decir, tiene una precisión del 50 % y un *recall* del 20 %. Con estos valores, el F1-score es:

$$F1_{score} = 2\,\frac{0.5 * 0.2}{0.5 + 0.2} = 2 * \frac{0.1}{0.7} = 0.29$$

Con el valor de F1 encontrado, es evidente que el modelo no funciona adecuadamente, dado que este valor está muy lejos de 1.

Miremos ahora este otro escenario:

		Predice 1	Predice 0
Real	1	7	3
	0	7	83

Calculemos de nuevo todas las métricas anteriores:

$$acc = \frac{7 + 83}{100} = \frac{90}{100} = 0.9$$

$$P = \frac{7}{7+7} = \frac{7}{14} = 0.5 P = \frac{7}{7+7} = \frac{7}{14} = 0.5, \quad \text{y} \quad R = \frac{7}{7+3} = \frac{7}{10} = 0.7$$

$$R = \frac{7}{7+3} = \frac{7}{10} = 0.7$$

$$F1_{score} = 2\,\frac{0.5 * 0.7}{0.5 + 0.7} = 2 * \frac{0.35}{1.2} = 0.58$$

Este modelo obtuvo la misma *acc* del primer modelo. No obstante, su F1-score es más alto. Adicionalmente, los valores de P y R están más equilibrados que en el primer caso, lo que evidencia un mejor compromiso entre P y R. Por lo que este modelo es muchísimo mejor en el escenario de de detección de cáncer, mejor que el anterior.

1.4. Model-centric vs. Data-centric

Hasta este punto del capítulo hemos recorrido, casi como si siguiéramos el pulso natural de un proyecto real, el ciclo de vida de la ciencia de datos: entendimos el contexto, analizamos el papel de los distintos perfiles técnicos, revisamos los tipos de problemas y conversamos sobre las métricas que permiten decir, con evidencia, qué tan bien se está comportando un modelo.

Ahora, nos detenemos en un cruce de caminos que aparece en casi todos los proyectos, aunque muchas veces pasa desapercibido: la elección entre centrar los esfuerzos en el modelo o centrar los esfuerzos en los datos. No es un dilema absoluto, sino una manera distinta de mirar la misma pregunta: *¿cómo logramos que un sistema funcione mejor?*

Durante años, la comunidad científica se inclinó con fuerza hacia el primer enfoque, el *model-centric*. Era natural: cada nueva arquitectura parecía prometer una revolución, un salto en precisión, un récord nuevo. Sin embargo, en los últimos años ha surgido una reflexión distinta y profundamente necesaria, encabezada por investigadores como Andrew Ng: *no existe modelo que brille si los datos que lo alimentan no son buenos*.

Dicho de otra forma: el desempeño no depende solo del ingenio del modelo, sino de la calidad del mundo que le mostramos a través de los datos.

Por eso, en esta sección vamos a explorar ambos enfoques, no como teorías aisladas, sino como dos lentes que nos permiten interpretar y mejorar un mismo sistema.

1.4.1. Model-centric

En el ciclo de vida que discutimos en la Sección 1.1, la etapa de modelado aparece casi al final. Y aun así (o quizás precisamente debido a eso) ha sido la etapa que más atención ha recibido históricamente. Aquí es donde los investigadores han propuesto arquitecturas nuevas, con capas que se conectan de maneras inesperadas, de redes capaces de “captar” patrones que antes parecían invisibles.

Por lo cual, cuando hablamos del enfoque centrado en el modelo, nos referimos a *ajustar sus hiperparámetros*, que típicamente se clasifican en:

- **Hiperparámetros de arquitectura:** incluye aumentar o disminuir la profundidad de la red, modificar la cantidad de filtros en una CNN, cambiar cómo se conectan las capas (secuencial, residual, paralela), o incluso probar familias completamente distintas como Transformers, ResNets o ConvNeXt. Cada una tiene una forma particular de interpretar la información, casi como si cada arquitectura *leyera el mismo texto* con un acento diferente.

- **Hiperparámetros de entrenamiento:** corresponden a cambios más sutiles, pero no menos poderosos como la tasa de aprendizaje, el optimizador, el número de épocas o el tamaño del *batch* (lote). También decidimos si el modelo debe aprender desde cero (como quien llega por primera vez a un terreno desconocido), o si utilizamos *transfer learning*, es decir, transferir conocimientos previos (en este caso los pesos) aprendidos en otro dataset.

La fortaleza del enfoque *model-centric* está en su disciplina comparativa: mantener quieta la entrada (los datos) y modificar únicamente la arquitectura y los hiperparámetros para medir cuánto cambian las métricas del modelo. Con este enfoque respondemos a la pregunta:

¿qué modelo es mejor para este problema, bajo estas condiciones?

Un ejemplo concreto de model-centric se encuentra en uno de los artículos de mi co-autoría titulado *Influence of Hyperparameters in Deep Learning Models for Coffee Rust Detection"* (Ver Figura 2).

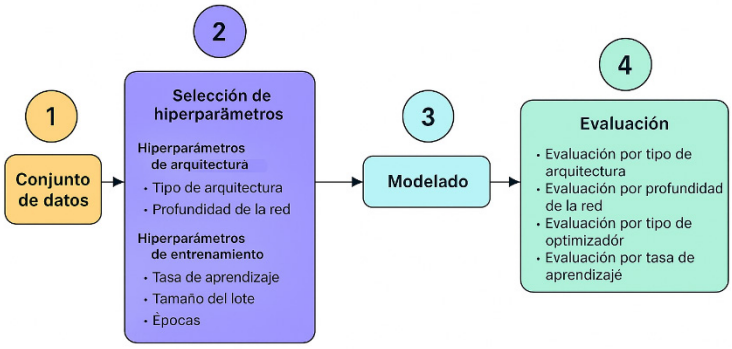


Figura 2. Ciclo de vida proyecto Coffee Rust Detection. Generada con ChatGPT por el autor.

Precisamente se evaluó el impacto de diferentes hiperparámetros como la profundidad de la red, el número de filtros, el tamaño de batch y la tasa de aprendizaje, en el desempeño del modelo al detectar roya del café. Fue un ejercicio casi quirúrgico: mantener constantes las imágenes y su preprocesamiento, y centrarnos en identificar qué combinaciones arquitectónicas y de entrenamiento producían la mayor capacidad de generalización. Ese estudio mostró, en un contexto agrícola real, cómo pequeñas variaciones en los hiperparámetros pueden transformar por completo la capacidad de un modelo para detectar enfermedades en cultivos. Por ejemplo, con la arquitectura DenseNet201 los valores de F1-score llegaron a ser un 40% mejores que los obtenidos con ResNet50, o que el optimizador podría hacer aumentar el F1 score en casi un 10% para la misma arquitectura; o variaciones del 5% cuando se cambiaba la tasa de aprendizaje y los demás hiperparámetros se mantenían fijos.

Es claro que este tipo de enfoque es importante dentro del ciclo de vida de un proyecto de datos, pero ahora, exploraremos en la siguiente sección por qué también el enfoque centrado en datos es necesario.

1.4.2. Data-centric

Mientras el enfoque *model-centric* enfoca sus esfuerzos y tiempo en las arquitecturas, el enfoque ***data-centric*** regresa al origen y vea a los datos como materia prima del proyecto. Aquí, antes de pensar en capas, hiperparámetros o métricas, aparece un gesto fundamental: comprender. Y esa comprensión empieza con un EDA (*Exploratory Data Analysis*) honesto y profundo, un recorrido inicial por la textura y el comportamiento de los datos, sean registros tabulares, series de tiempo con pulsos y estacionalidades, o señales más complejas como el audio, para descubrir qué cuentan y qué ocultan.

Desde esta mirada, el *dataset* deja de ser un simple insumo para el entrenamiento y se convierte en el terreno donde la solución realmente se construye. El propósito ya no es entrenar un modelo, sino *representar un fenómeno*, y eso exige que los datos representen ese fenómeno con claridad.

Por eso, el trabajo *data-centric* pone especial cuidado en la ingeniería de características (*feature engineering*) cuyo propósito principal se centra en transformar datos crudos creando nuevas variables que capturen patrones invisibles, transformando aquellas que necesitan una escala común, estandarizando representaciones y completando valores faltantes de manera coherente con el contexto. Es un proceso creativo y técnico a la vez: una mezcla entre intuición, conocimiento del dominio y aplicación de buenas prácticas.

Lo que se busca es simple de decir, pero profundo en su trasfondo: que los datos transformados permitan que el modelo aprenda mejor.

Cuando esta etapa se trabaja con cuidado, el efecto es tangible: modelos más estables, comportamientos más consistentes, métricas que dejan de oscilar caprichosamente y una comprensión más fina del problema real. Incluso arquitecturas sencillas pueden alcanzar un desempeño notable cuando están apoyadas en un *dataset* bien construido, bien entendido y bien acondicionado.

El *data-centric* no es un complemento, es la base. Es la parte del proceso que nos recuerda que, antes de afinar un modelo, debemos asegurarnos de que los datos le hablen con claridad. Y cuando lo hacen, el aprendizaje se vuelve más eficiente, más confiable y cercano a la realidad que intentamos representar.

1.4.3. Enfoque híbrido

En la práctica, pocos proyectos se desarrollan bajo un enfoque exclusivamente model-centric o puramente data-centric. La mayoría de los procesos reales (esos que involucran datos imperfectos, restricciones de tiempo y objetivos cambiantes), requieren un enfoque **híbrido**, capaz de integrar decisiones tanto sobre la arquitectura como sobre los datos.

Este enfoque reconoce que el desempeño de un modelo no depende de un solo factor, sino de la interacción entre varios elementos: la calidad y profundidad del dataset, la riqueza de las características diseñadas, la forma en que se representan las señales, y la capacidad del modelo para capturar relaciones complejas. En un proyecto híbrido, no se parte de una única respuesta, sino de una pregunta constante:

¿Dónde podremos impactar de mejor manera, transformando datos o modificando los hiperparámetros del modelo?

A veces, una transformación adecuada en los datos puede desbloquear mejoras significativas. Otras veces, es la arquitectura la que necesita ajustarse, ya sea agregando capas, probando una red preentrenada, cambiando la función de activación, o incluso replanteando el tipo de modelo usado.

La riqueza del enfoque híbrido está en esa flexibilidad: en la capacidad del equipo para iterar entre los datos y el modelo, sin casarse con una sola dimensión del problema. No se trata de mezclar al azar, sino de establecer ciclos de experimentación bien definidos, donde cada modificación en los datos se prueba con diferentes modelos, y cada nuevo modelo se evalúa bajo condiciones de datos controladas y documentadas.

Un ejemplo destacado de este enfoque es el proyecto **FakeVoiceFinder**, en el cual he venido trabajando desde 2025. Es un *framework* en Python, publicado en GitHub, que permite realizar análisis *data-centric* y *model-centric* para la detección de audio sintético generado con IA. FakeVoiceFinder cubre casi todo el ciclo de vida del proyecto y solo necesita un insumo inicial: un dataset de audios naturales y sintéticos etiquetados. A partir de allí, automatiza el resto del proceso: partición de datos, ajuste de longitud de los audios, generación de representaciones espectrales, ajuste de hiperparámetros, selección y

entrenamiento de arquitecturas, cálculo de métricas y visualización comparativa de resultados. Desde la perspectiva del usuario, su uso es sencillo: puede seleccionar los valores de cada etapa o, si lo prefiere, dejar que el *framework* ejecute todo con los parámetros predeterminados.

Con esta estructura, el usuario puede comparar el desempeño de todas las arquitecturas para una misma representación espectral o, en sentido inverso, evaluar las cuatro representaciones utilizando una arquitectura específica. Además, *FakeVoiceFinder* permite analizar ambos elementos de manera conjunta, de modo que es posible identificar qué modelos son más sensibles al tipo de transformación y qué representaciones se ajustan mejor a cada arquitectura, revelando relaciones que no serían evidentes desde un único enfoque.

Por ello, el enfoque híbrido no es solamente una estrategia técnica; es también una actitud metodológica. Implica mirar el proyecto con humildad, evitar suponer que un solo componente determina el éxito o el fracaso, y mantener siempre una lógica experimental, comparativa y reproducible. Es, quizás, la forma más realista y más poderosa de abordar proyectos complejos, en los que los datos y los modelos se conciben y se refinan juntos, como partes de un mismo sistema vivo.