santi.cornelio

PLAYBOOK · DATA SCIENCE

Playbook ilustrativo de Data Science

Qué aprender primero, qué después, y qué te puedes saltar sin culpa.

Qué es esto (y qué no)

Qué es: un mapa de dependencias para aprender Data Science desde cero: qué va antes de qué y por qué, organizado en siete etapas, cada una con un proyecto de comprobación real, no un ejercicio de tutorial.

Qué no es:

- No es un curso. No tiene clases ni ejercicios evaluados: para eso, cada etapa enlaza a los cursos y recursos ya verificados en el catálogo del sitio.
- No es una promesa de fecha fija. Nadie puede prometerte cuándo vas a terminar: depende de cuánto programes ya y de cuántas horas por semana le dediques.
- No es la lista de todo lo que existe en Data Science. Es un filtro: lo que de verdad necesitas, en el orden en que lo necesitas, y, tan importante como lo anterior, lo que te puedes saltar según tu objetivo.

Cuánto tarda de verdad: entre 220 y 370 horas de trabajo deliberado repartidas en las siete etapas. Con 5 horas por semana, más de un año. Con 20 horas por semana, unos 3-4 meses. El detalle etapa por etapa está más abajo, en "Cómo leer este playbook".

Quién lo escribió: Santiago Cornelio explica machine learning y data science aplicados a negocios, en español, en `@santi.cornelio`. Trabaja como consultor de datos en Barcelona, y este playbook es el orden en el que a él le hubiera gustado aprenderlo, no la lista de todo lo que existe.

Cómo leer este playbook

Tiempo real, no una promesa. Sumando las siete etapas con dedicación constante, esto son aproximadamente 220 a 370 horas de trabajo deliberado (no de calendario: una persona que le dedica 5 horas por semana tarda bastante más de un año en llegar al final; una que le dedica 20 horas por semana, unos 3-4 meses). La variación depende casi por completo de si ya programas antes de empezar. Nadie te puede prometer una fecha fija.

No todas las etapas son para todos. Este playbook asume el objetivo más completo: llegar a construir y desplegar un modelo. Si tu objetivo es otro, hay partes enteras que te puedes saltar:

- **Si eres dueño de negocio y no vas a programar:** este playbook técnico no es para ti. Mira en su lugar Cisco Networking Academy — Introduction to Data Science o Elements of AI, pensados para entender qué se puede hacer con datos sin escribir código.
- **Si tu objetivo es un rol de analista de datos:** puedes parar después de la **Etapa 4 (Visualización y comunicación)**. Las etapas 5 y 6 (Machine Learning y producción) no son requisito para ese rol: son valiosas, pero no obligatorias.
- **Si tu objetivo es un rol de Data Scientist o ML Engineer:** necesitas las siete etapas completas. No hay atajo real para SQL o estadística cuando el trabajo es con datos de producción: son la parte que más se nota cuando falta.

Por qué este orden. Python antes que pandas porque pandas es ilegible si no entiendes listas, diccionarios y funciones primero. SQL después de pandas, no antes ni en paralelo, porque para entonces ya sabes qué es una tabla y un `groupby`, y SQL se vuelve "lo mismo con otra sintaxis" en vez de un lenguaje nuevo desde cero. Estadística después de manipulación de datos porque necesitas tener datos limpios en la mano antes de que una prueba de hipótesis signifique algo. Visualización antes de ML a propósito: comunicar bien un hallazgo simple es más valioso en la práctica que un modelo que nadie entiende. Y ML al final porque es la etapa que más depende de todas las anteriores: un modelo construido sobre datos sucios o sin validación estadística es ruido con forma de número.

MAPA DE DEPENDENCIAS

Las siete etapas, en orden

Cada etapa depende de que la anterior esté resuelta. No es la única secuencia posible: es la que menos fricción tiene.

ETAPA 0

Fundamentos de programación (Python)

↓

ETAPA 1

Manipulación de datos (pandas y NumPy)

↓

ETAPA 2

SQL

↓

ETAPA 3

Estadística y probabilidad

↓

ETAPA 4

Visualización y comunicación

Suficiente si tu meta es un rol de analista de datos.

↓

ETAPA 5

Machine Learning

↓

ETAPA 6

Puesta en producción

⟨ ⟩ Etapa 0: Fundamentos de programación (Python)

Qué puedes hacer al terminar: escribir un script propio que lee un archivo, transforma su contenido con listas y diccionarios, y automatiza una tarea repetitiva, sin depender todavía de ninguna librería de datos.

Conceptos que importan:

- Tipos de datos y estructuras (listas, diccionarios, tuplas, sets).
- Control de flujo (condicionales, loops, comprensión de listas).
- Funciones, parámetros y alcance de variables.
- Manejo de archivos y manejo básico de excepciones (`try` / `except`).

Lo que te puedes saltar: programación orientada a objetos avanzada (herencia múltiple, metaclases, decoradores propios). Ocupa buena parte de cualquier curso de "Python completo", pero no la vas a necesitar hasta que un proyecto real te lo pida. Cuando llegue ese momento la vas a entender mejor con ese contexto encima que memorizada de antemano, sin haberla usado nunca.

Recursos:

- Harvard CS50x — Introduction to Computer Science (la parte de Python; cero absoluto).
- Kaggle Learn: micro-curso de Python, para practicar en notebook sin instalar nada.

Proyecto de comprobación: descarga un CSV pequeño (menos de 5 MB) de datos.gob.es y, sin usar pandas todavía, escribe un script que lo lea línea por línea y calcule un resumen (conteos por categoría, promedio de una columna numérica, el valor más frecuente) usando solo estructuras nativas de Python. Forzarte a hacerlo sin pandas es lo que revela si entendiste listas y diccionarios o solo los reconoces de memoria.

Error típico de esta etapa: pasar semanas en sintaxis sin escribir un programa completo propio, o saltar directo a pandas sin tener resuelto lo anterior. Cuando después aparece un error de pandas no hay con qué razonarlo, porque la base de estructuras de datos nunca se afianzó.

Etapa 1: Manipulación de datos (pandas y NumPy)

Qué puedes hacer al terminar: tomar un archivo crudo y desordenado (con nulos, duplicados, tipos mezclados y fechas inconsistentes) y dejarlo limpio, filtrado y combinado con otra fuente, listo para analizar.

Conceptos que importan:

- DataFrames y Series: creación, inspección, selección (`loc` / `iloc`, indexado booleano).
- Combinar datos: `merge`, `join`, `concat`, y cuándo usar cada uno.
- `groupby` y agregaciones (`sum`, `mean`, `agg` con múltiples funciones).
- Fechas: parsing, extracción de componentes, `resample` para series de tiempo.

Lo que te puedes saltar: optimización de memoria y tipos de dato (`category`, downcasting de enteros/floats) para datasets de decenas de millones de filas. Aparece en cualquier curso avanzado de pandas, pero solo importa cuando el dataset literalmente no entra en RAM, y eso no pasa en ninguno de los proyectos de este playbook.

Recursos:

- Corey Schafer — Pandas Tutorials: la referencia en profundidad de pandas.
- Python Data Science Handbook (capítulos de NumPy y Pandas): para entender el porqué detrás de las operaciones vectorizadas.
- freeCodeCamp — Data Analysis with Python: si quieres un certificado con proyectos evaluados automáticamente.

Proyecto de comprobación: limpia y transforma el dataset UCI — Online Retail (tiene devoluciones como cantidades negativas, `CustomerID` faltante en buena parte de las filas, y fechas que necesitas parsear) y construye una tabla de "valor de compra por cliente y por mes". No es un dataset arreglado de antemano para que el ejercicio salga limpio: tienes que decidir qué hacer con los nulos y las devoluciones, que es exactamente el trabajo real.

Error típico de esta etapa: iterar con un `for` fila por fila sobre un DataFrame en vez de usar operaciones vectorizadas. Funciona sin problema en un dataset de mil filas y se vuelve inviable, o se cuelga directamente, en cuanto el dataset crece. Es la señal más clara de haber aprendido la sintaxis de pandas sin aprender cómo piensa pandas.

Etapa 2: SQL

Qué puedes hacer al terminar: extraer exactamente los datos que necesitas de una base relacional, sin depender de que alguien más te pase un CSV ya armado.

Conceptos que importan:

- `SELECT`, `WHERE`, `ORDER BY` y filtrado básico.
- `JOIN`s (inner, left) y cuándo cada tipo cambia el resultado.
- `GROUP BY` con `HAVING`, y la diferencia con filtrar antes de agrupar.
- Subqueries, CTEs (`WITH`) y *window functions* (`ROW_NUMBER`, `RANK`, funciones con `OVER`).

Lo que te puedes saltar: administración de bases de datos: usuarios, permisos, tuning de índices, replicación. Es trabajo de un DBA o de ingeniería de datos, no de quien analiza o modela con datos. Saber que existe basta por ahora.

Recursos:

- Alex The Analyst — "SQL Full Course: Beginner to Advanced": de sintaxis básica hasta window functions.
- Google BigQuery — Public Datasets: para practicar con datos reales a escala, sin instalar un motor de base de datos.
- DBeaver Community Edition: cliente para conectarte a una base de datos propia si prefieres practicar fuera de BigQuery.

Proyecto de comprobación: usando los datasets públicos de BigQuery (taxis de Nueva York o actividad pública de GitHub), escribe una consulta con una *window function* que responda una pregunta de negocio concreta, por ejemplo "¿qué hora del día tiene más viajes por zona de la ciudad?" o "¿qué repositorios tuvieron más actividad por semana en el último mes?". No un `SELECT *` de práctica: la pregunta tiene que obligarte a usar `PARTITION BY`.

Error típico de esta etapa: resolver con subqueries anidadas en tres niveles lo que una *window function* hace en una sola línea. El resultado es correcto, pero es ilegible, lento, y es la señal más clara de no haber entendido todavía qué resuelven las window functions.

△ Etapa 3: Estadística y probabilidad

Qué puedes hacer al terminar: mirar una diferencia entre dos números y decidir, con criterio, si es real o es ruido de muestra, antes de escribir una conclusión en un reporte.

Conceptos que importan:

- Distribuciones y variables aleatorias (qué forma tienen los datos y por qué importa).
- Intervalos de confianza: qué tan seguro puedes estar de un número.
- Pruebas de hipótesis (t-test, chi-cuadrado) y qué pregunta responde cada una.
- Correlación vs. causalidad: por qué una no implica la otra.

Lo que te puedes saltar: inferencia bayesiana computacional avanzada (MCMC, modelos jerárquicos). Es territorio de investigación o de problemas muy específicos; no es necesario para analizar datos de negocio con criterio estadístico sólido, que es el objetivo de esta etapa.

Recursos:

- Think Stats, 2ª edición: estadística aplicada con código, no solo fórmulas.
- StatQuest with Josh Starmer — "Statistics Fundamentals": para construir intuición visual de cada concepto antes o después del libro.
- MIT OpenCourseWare — 18.05: si quieres rigor matemático completo, con problem sets y soluciones.

Proyecto de comprobación: retoma el dataset de Online Retail que ya limpiaste en la Etapa 1 y responde con una prueba de hipótesis (no a ojo) si la diferencia de gasto promedio por compra entre dos países del dataset es estadísticamente significativa o es ruido de muestra, dado el tamaño de cada grupo.

Error típico de esta etapa: reportar una diferencia entre dos grupos como "hallazgo" solo porque los promedios se ven distintos, sin revisar el tamaño de muestra ni calcular un intervalo de confianza. Es la forma más común de *p-hacking* involuntario, y la que más rápido le quita credibilidad a un análisis frente a alguien que sí sabe estadística.

Etapa 4: Visualización y comunicación

Qué puedes hacer al terminar: armar un gráfico o un dashboard que una persona sin formación técnica entiende sin que tengas que explicárselo en voz.

Conceptos que importan:

- Elegir el tipo de gráfico según la pregunta que respondes, no por costumbre o porque "así se ve bien".
- Jerarquía visual: qué resaltar y qué directamente omitir del gráfico.
- Dashboards interactivos vs. gráficos estáticos: cuándo cada uno comunica mejor.
- Contar una conclusión por gráfico. Un gráfico con diez variables no comunica diez cosas, no comunica nada.

Lo que te puedes saltar: dominar de punta a punta una herramienta de BI corporativa completa (Power BI o Tableau). Para este propósito, un dashboard bien pensado en Streamlit comunica exactamente igual, y no depende de una licencia ni de Windows.

Recursos:

- Python Data Science Handbook (capítulo de visualización con Matplotlib): principios de diseño de gráficos, no solo sintaxis.
- Streamlit Community Cloud: para pasar de un gráfico en un notebook a un dashboard con una URL pública.

Proyecto de comprobación: con el dataset Kaggle — Spotify Tracks Dataset, construye un dashboard en Streamlit que responda **una sola** pregunta clara, por ejemplo "¿qué hace que una canción sea bailable?", con 3 o 4 gráficos curados para esa pregunta específica, no un explorador genérico con un filtro por cada columna del dataset.

Error típico de esta etapa: construir un dashboard con un control o filtro para cada columna disponible, en vez de decidir de antemano qué pregunta se está respondiendo. Se ve "completo" y da la sensación de más trabajo, pero no comunica nada porque no tiene un punto.

Etapa 5: Machine Learning

Qué puedes hacer al terminar: construir un modelo predictivo, validarlo de forma correcta, y explicar con evidencia si de verdad funciona o solo lo parece.

Conceptos que importan:

- `train` / `test` split y validación cruzada, y por qué evaluar sobre los mismos datos de entrenamiento no cuenta.
- Overfitting vs. underfitting, y cómo se ven en la práctica (no solo en la gráfica del tutorial).
- Elegir la métrica según el problema: *accuracy* casi nunca es la métrica correcta cuando las clases están desbalanceadas.
- Reconocer si el problema real es regresión, clasificación o series de tiempo, antes de elegir el algoritmo.

Lo que te puedes saltar: arquitecturas de deep learning de última generación (transformers, modelos de difusión) si el objetivo son problemas de negocio con datos tabulares. En ese terreno, un modelo de árboles (random forest, gradient boosting) bien validado le gana a una red neuronal la mayoría de las veces, y es muchísimo más simple de explicar y de mantener.

Recursos:

- An Introduction to Statistical Learning (ISLR / ISLP): la base conceptual, con versión en Python (ISLP).
- Stanford CS229: si quieres entender la derivación matemática detrás de los algoritmos, no solo usarlos.
- freeCodeCamp — Machine Learning with Python: para implementar con proyectos evaluados y certificado real.

Proyecto de comprobación: con Kaggle — Rossmann Store Sales, construye un modelo que prediga ventas diarias por tienda incorporando estacionalidad, feriados y promociones, y válidalo con una partición **temporal** de los datos (entrenar con el pasado, validar con el futuro), no con un split aleatorio, porque esto es una serie de tiempo.

Error típico de esta etapa: hacer un split aleatorio de entrenamiento/prueba sobre datos que tienen orden temporal, como ventas diarias. El modelo termina "viendo el futuro" durante el entrenamiento sin que te des cuenta, y la métrica de validación se ve excelente mientras miente sobre qué tan bien funcionaría el modelo en producción.

Etapa 6: Puesta en producción

Qué puedes hacer al terminar: dejar un modelo o un análisis funcionando en una URL que otra persona puede usar, sin que necesite tener Python instalado ni saber qué es un notebook.

Conceptos que importan:

- Separar entrenamiento de inferencia: el modelo se entrena una vez, no en cada solicitud.
- Empaquetar dependencias de forma reproducible (`requirements.txt` o equivalente) para que la app no dependa de "lo que tengo instalado en mi máquina".
- Servir el modelo detrás de una interfaz simple o una API mínima.
- Monitoreo básico: si llegan datos nuevos con el tiempo, ¿cómo te enteras de que el modelo dejó de funcionar bien?

Lo que te puedes saltar: orquestación de MLOps compleja (Airflow, Kubernetes, pipelines de CI/CD completos). Es necesario en equipos grandes con muchos modelos en producción simultánea. No lo es para el primer proyecto que alguien despliega en solitario, y aprenderlo sin ese contexto es memorizar herramientas sin saber para qué problema existen.

Recursos:

- Streamlit Community Cloud o Hugging Face Spaces: para desplegar la app sin administrar un servidor.
- Supabase: si el proyecto necesita guardar datos o resultados de usuarios, no solo servir un modelo entrenado.

Proyecto de comprobación: toma el modelo de forecasting de Rossmann de la Etapa 5 y expónlo en una app de Streamlit desplegada en Streamlit Community Cloud, donde alguien ingresa una tienda y una fecha y recibe una predicción de ventas. Esto cierra el ciclo completo: dato crudo → limpieza → análisis → modelo → producto que otra persona puede abrir.

Error típico de esta etapa: dejar el modelo entrenado solo en un notebook en tu máquina. "Funciona en mi máquina" no es un producto: nadie más lo puede usar, y en un portafolio no demuestra nada que un screenshot no demuestre igual de bien (es decir, casi nada).

Sigue desde aquí

Este playbook llega hasta un primer proyecto completo y desplegado, no hasta el límite del campo. Después de la Etapa 6 hay especialización: NLP, visión por computadora, series de tiempo avanzadas, ingeniería de datos. Cuál de esas rutas tomar depende del problema que termines queriendo resolver, así que no tiene sentido prescribirla aquí antes de que la elijas.

El catálogo completo en el sitio tiene la ficha de cada recurso que este playbook solo enlaza, con fecha de última verificación y una nota de para quién NO sirve cada uno. Se actualiza aparte de este PDF: si un enlace se rompe, se arregla ahí, no en una nueva versión de este documento.

Para que te avise cuando esta guía se actualice o salga la siguiente pieza, suscríbete en `santi-cornelio.xyz` ; la newsletter está debajo del catálogo.

`@santi.cornelio` en Instagram y TikTok.