



Claude Code para Ciencia de Datos

Dr. Carlos Ricardo Cruz Mendoza Dr. Noé Hernández

IIMAS · IA para Ciencias de Datos

8 de septiembre de 2026

Introducción a Claude Code

- Qué es y cómo funciona el ciclo del agente
- Herramientas, contexto y memoria
- Permisos y cómo deshacer
- Instalación y comandos
- Prompts

Ejercicios

Bloque 1 · Datos

1. Análisis exploratorio (EDA)
2. Limpieza de datos
3. Visualización y PowerPoint
4. SQL en lenguaje natural

Bloque 2 · Obtención de datos

5. Integración con una API
6. Web scraping

Bloque 3 · Modelado

7. Ingeniería de características
8. Prueba A/B y estadística
9. Pipeline de ML de punta a punta
10. Aplicación de Streamlit

Cierre



¿Qué es Claude Cowork y Claude Code?

Ambos son Agentes que utilizan modelos de Antropic que viven en tu terminal/escritorio, con acceso a los archivos de tu proyecto, a tu shell y a la web, y que trabajan de forma iterativa hasta terminar la tarea. La diferencia principal es que Claude Cowork es un agente de escritorio para usuarios no técnicos enfocado en organizar carpetas y automatizar tareas de oficina, mientras que Claude Code es una herramienta avanzada basada en la terminal orientada a desarrolladores para escribir y gestionar software.

Dónde se usa

- Terminal, aplicación de escritorio e IDEs como VS Code y JetBrains.
- Requiere una cuenta de Claude (Pro, Max, Team o Enterprise) o de Claude Console.



Cómo funciona: el ciclo de agente

Tres fases que se repiten

1. **Reúne contexto:** busca y lee archivos, ejecuta comandos para observar el estado.
2. **Actúa:** edita archivos, corre scripts, consulta la web.
3. **Verifica:** ejecuta pruebas, revisa la salida y se corrige.

Repite hasta terminar, encadenando decenas de acciones y ajustando el rumbo con lo que aprende en cada paso.

«Arregla las pruebas que fallan»

1. Corre la suite de pruebas.
2. Lee la salida de error.
3. Busca los archivos implicados.
4. Los lee para entender el código.
5. Edita lo necesario.
6. Vuelve a correr las pruebas.

Si presionas la tecla Esc mientras se está ejecutando una acción se detiene al instante; si en vez de eso escribes una corrección y das Enter, la lee al terminar la acción en curso y ajusta el rumbo sin detenerse.



Instalación y comandos de terminal

Instalar

<code>curl -fsSL https://claude.ai/install.sh bash</code>	macOS, Linux y WSL
<code>irm https://claude.ai/install.ps1 iex</code>	Windows PowerShell
<code>brew install --cask claude-code</code>	Homebrew
<code>winget install Anthropic.ClaudeCode</code>	WinGet

Usar

<code>claude --version</code>	confirma que quedó instalado
<code>claude</code>	abre una sesión interactiva en la carpeta actual
<code>claude "limpia el csv de clientes"</code>	abre la sesión con una petición inicial
<code>claude -p "explica este script"</code>	responde una sola vez y termina
<code>claude -c</code>	continúa la conversación más reciente de esta carpeta
<code>claude -r</code>	retoma una conversación anterior
<code>claude --model opus</code>	elige el modelo de la sesión
<code>claude --permission-mode plan</code>	arranca en modo plan, sin editar nada
<code>claude --add-dir ../datos</code>	le da acceso a otra carpeta
<code>claude doctor</code>	diagnostica la instalación



Las herramientas y lo que alcanza a ver

Herramientas integradas

Archivos	leer, editar, crear y reorganizar
Búsqueda	encontrar por patrón o por contenido
Ejecución	shell, git, pruebas, servidores
Web	buscar y traer documentación
Código	errores de tipo, ir a la definición

A qué tiene acceso

- Los archivos de tu carpeta de trabajo, y otras si lo autorizas.
- Tu terminal: cualquier comando que tú podrías correr.
- El estado de git: rama, cambios sin confirmar, historial.
- CLAUDE.md: las instrucciones y convenciones del proyecto, que lee al iniciar cada sesión.
- Extensiones: servidores MCP, *skills* y subagentes.

Claude elige qué herramienta usar según tu petición y según lo que va descubriendo.



Contexto, memoria y sesiones

La ventana de contexto

- Guarda la conversación, los archivos leídos, las salidas de comandos y el `CLAUDE.md`.
- Se llena. Al llenarse, Claude Code descarta salidas viejas y resume la conversación.
- Al resumir se pueden perder las instrucciones del principio.

Por eso las reglas permanentes van en `CLAUDE.md`, no en la conversación.

Qué hacer al respecto

- `/context` muestra qué está ocupando espacio.
- `/compact` resume y libera espacio; puedes indicarle en qué enfocarse.
- `/clear` arranca de cero entre tareas que no tienen relación.
- Cada sesión empieza con contexto nuevo, pero queda guardada: `claude -c` o `/resume` la retoman.

Regla práctica. Si algo debe ser cierto durante toda la sesión —convenciones de código, rutas, paleta de colores, formato de salida— va en `CLAUDE.md`. Si solo aplica a esta petición, va en el prompt.



Permisos y cómo deshacer

Modos de permiso — Shift+Tab los cicla

Auto	un clasificador revisa las acciones y bloquea las riesgosas
Manual	pregunta antes de cada edición y cada comando
Aceptar ediciones	edita y usa comandos comunes de archivos sin preguntar
Plan	explora y propone, sin tocar tus archivos

Deshacer

- Antes de editar un archivo guarda una copia del contenido.
- Esc Esc abre el menú para retroceder código y conversación.
- Puedes ocupar el comando `/rewind`

Lo que no se puede deshacer: lo que toca sistemas externos — bases de datos, APIs, despliegues. Eso se controla con el modo de permisos y con reglas por comando (`/permissions`).



Comandos dentro de la sesión

<code>/help</code>	lista los comandos
<code>/init</code>	genera el <code>CLAUDE.md</code> del proyecto
<code>/memory</code>	edita el <code>CLAUDE.md</code>
<code>/model</code>	cambia de modelo
<code>/config</code>	ajustes: tema, modelo, editor
<code>/permissions</code>	permisos por comando
<code>/mcp</code>	servicios externos (MCP)
<code>/cost</code> o <code>/usage</code>	tokens y gasto de la sesión

<code>/context</code>	qué ocupa el contexto
<code>/compact</code>	resume y libera contexto
<code>/clear</code>	conversación nueva
<code>/plan</code>	entra en modo plan
<code>/review</code>	revisa los cambios
<code>/resume</code>	vuelve a una sesión previa
<code>/rewind</code>	retrocede a un punto previo
<code>/exit</code>	sale de Claude Code

Atajos. `/` lista los comandos • Tab autocompleta • ↑ historial • Shift+Tab cambia el modo de permisos • Esc interrumpe • @archivo referencia un archivo en tu petición



Cómo pedir bien las cosas

Un prompt vago

«analiza estos datos»

Un prompt que sirve

«lee `ventas.csv`, calcula la conversión por región del cuarto trimestre, exporta el resultado a `resultados.csv` con dos decimales y valida que no haya regiones faltantes»

Cuatro hábitos

- **Deja que explore primero.** El modo plan lee sin tocar nada y te propone un plan.
- **Dale una forma de verificar:** una prueba, un total conocido, una validación.
- **Revisa el código antes de aceptarlo,** no solo el resumen final.
- **Conserva el script,** no solo el resultado: eso es lo que puedes volver a correr.



La idea

- Realizar 10 tareas con Claude Code.
- En cada una: el enunciado, el conjunto de datos y el *prompt* listo para copiar.
- El objetivo no es el resultado, es el flujo de trabajo.

Tu tarea: decidir qué preguntar, revisar lo que devuelve y detectar cuándo el resultado *parece* correcto pero no lo es.

Regla

Ningún resultado se acepta sin verificarlo contra los datos.



Ejercicio 1 · Análisis exploratorio de datos (EDA)

Objetivo

Obtener en minutos el panorama completo de un conjunto de ventas: forma, tipos, nulos, distribuciones, correlaciones y problemas de calidad.

Datos de entrada

`ventas_ecommerce.csv` — 260 filas $\times$ 11 columnas

Qué debes tener presente

- El archivo trae 68 valores nulos repartidos entre descuento, ingreso y cantidad.
- Hay un pedido con ingreso negativo y varios pedidos cancelados o pendientes.
- Los ingresos están sesgados a la derecha: la media y la mediana no coinciden.

Columnas: `id_pedido`, `id_cliente`, `nombre_cliente`, `producto`, `categoria`, `region`, `fecha_pedido`, `ingreso`, `cantidad`, `descuento`, `estatus`

Entregables: Script `eda_ventas.py` reejecutable y comentado • Al menos 4 gráficas en PNG • Resumen escrito con los hallazgos y las banderas de calidad



Ejercicio 1 · el prompt

Prompt para Claude Code

Tengo un conjunto de datos de ventas de comercio electrónico llamado `ventas_ecommerce.csv` que está en esta carpeta. Ejecuta un análisis exploratorio de datos completo. Quiero ver:

1. Dimensiones del conjunto, tipos de dato y un resumen de valores nulos.
2. Estadística descriptiva de las variables numéricas.
3. Gráficas de distribución de ingreso, cantidad y descuento.
4. Mapa de calor de correlaciones.
5. Los 5 productos con más ingreso y el desglose por categoría y por región.
6. Cualquier valor atípico o problema de calidad de datos que detectes.

Guarda todas las gráficas como PNG y escríbeme un resumen con los hallazgos clave.



Ejercicio 2 · Limpieza de datos

Objetivo

Convertir un archivo heredado, con formatos inconsistentes y duplicados, en una tabla confiable. Es el trabajo que normalmente toma de 30 minutos a 2 horas de código pandas lleno de casos borde.

Datos de entrada

`clientes_sucios.csv` — 100 filas × 9
columnas (49 son duplicados)

Qué debes tener presente

- Nombres con mayúsculas mezcladas, teléfonos en 5 formatos, fechas en 5 formatos y países en varias grafías.
- Ingresos como texto con \$ y comas; edades vacías o imposibles; activo con yes, 1, true, no, 0, False.
- Los duplicados no están contiguos: hay que detectarlos por contenido.

Columnas: `id_cliente`, `nombre_cliente`, `email`, `telefono`, `fecha_registro`, `ingreso_anual`, `pais`, `edad`, `activo`

Entregables: `clientes_limpios.csv` (51 filas) con validación de correo • Script `limpieza_clientes.py` • Comparativa antes/después



Ejercicio 2 · el prompt

Prompt para Claude Code

Tengo un conjunto de datos de clientes muy sucio en `clientes_sucios.csv`. Límpialo por completo:

1. Estandariza `nombre_cliente` a formato Título.
2. Valida el formato de `email` y marca los inválidos en una columna nueva.
3. Estandariza `telefono` al formato (555) 123-4567.
4. Convierte todos los valores de `fecha_registro` al formato AAAA-MM-DD.
5. Limpia `ingreso_anual`: quita el \$ y las comas y conviértelo a número decimal.
6. Estandariza `pais` al nombre completo del país.
7. Corrige `edad`: elimina edades imposibles (menores que 0 o mayores que 100).
8. Estandariza `activo` a valores booleanos verdadero/falso.
9. Elimina las filas duplicadas.

Muéstrame una comparación antes/después con el número de filas y columnas, y guarda el resultado como `clientes_limpios.csv`.

Punto de atención. Revisa a mano dos o tres duplicados detectados: ¿de verdad son la misma persona? Un criterio de deduplicación demasiado laxo borra clientes reales.



Ejercicio 3 · Visualización y PowerPoint automático

Objetivo

Pasar del dato a la presentación sin cambiar de herramienta: cargar los datos, generar gráficas con calidad de publicación y armar el deck completo con interpretaciones, todo desde un solo prompt.

Datos de entrada

`ingresos_mensuales.csv` — 48 filas (12 meses $\times$ 4 regiones)

Qué debes tener presente

- El objetivo real es un reporte que se repite cada mes: si el 80–90 % es idéntico, conviene automatizarlo.
- Un archivo `CLAUDE.md` en la carpeta permite fijar paleta, tipografía y formato una sola vez.
- La ventaja frente a generar el deck a mano es que queda un script de Python reejecutable.

Columnas: `mes`, `nombre_mes`, `region`, `ingreso`, `unidades_vendidas`, `valor_promedio_pedido`, `clientes_nuevos`

Entregables: 4 gráficas PNG con estilo consistente • `ingresos.pptx` con 6 diapositivas • Script `reporte_ingresos.py` reejecutable el mes siguiente



Ejercicio 3 · el prompt

Prompt para Claude Code

Tengo un conjunto de datos de ingresos mensuales llamado `ingresos_mensuales.csv`. Crea las siguientes visualizaciones:

1. Una gráfica de líneas del ingreso por mes.
2. Una gráfica de barras del ingreso total por región.
3. Un mapa de calor de mes contra región.
4. Una gráfica resumen con los indicadores principales.

Que las gráficas queden limpias y listas para presentación, con títulos, etiquetas y una paleta de colores consistente; guárdalas como PNG. Después crea una presentación de PowerPoint con: una diapositiva de título «Desempeño de ingresos 2024»; una diapositiva por gráfica con 2 o 3 oraciones de interpretación debajo de cada una; y una diapositiva final con los 3 hallazgos principales. Guarda la presentación como `ingresos.pptx`.



Ejercicio 4 · SQL en lenguaje natural

Objetivo

Describir en español lo que se quiere saber y obtener el SQL, la ejecución y los resultados. Especialmente útil con joins complejos, CTE y manejo de fechas.

Datos de entrada

sql_clientes.csv (120),
sql_pedidos.csv (620),
sql_productos.csv (12)

Qué debes tener presente

- No hace falta una base de datos: se cargan los CSV en SQLite en memoria.
- Conviene describir las tablas y las columnas relevantes dentro del propio prompt.
- `nivel` toma los valores `bronce`, `plata` y `oro`.

Columnas: clientes: `id_cliente`, `nombre`, `nivel`, `pais`, `fecha_alta` · pedidos: `id_pedido`, `id_cliente`, `id_producto`, `fecha_pedido`, `cantidad`, `monto`, `estatus`, `region` · productos: `id_producto`, `nombre_producto`, `categoria`, `precio`, `costo`

Entregables: Script `consultas_sql.py` con las 5 consultas comentadas • Resultados de cada consulta • Nota explícita sobre cualquier supuesto que haya tomado el modelo



Ejercicio 4 · el prompt

Prompt para Claude Code

Tengo tres archivos CSV: `sql_clientes.csv`, `sql_pedidos.csv` y `sql_productos.csv`. Cárgalos en una base de datos SQLite en memoria y escribe consultas SQL para responder lo siguiente:

1. Ingreso total por nivel de cliente.
2. Los 10 clientes que más han gastado en los últimos 6 meses.
3. El crecimiento del ingreso mes contra mes.
4. Los clientes con más de 3 pedidos que nunca han cancelado.
5. El valor promedio del pedido por región.

Para cada consulta muéstrame el SQL, ejecútalo y despliega los resultados. Si algún dato que pido no existe en las tablas, dímelo en lugar de inventarlo.



Ejercicio 5 · Integración con una API

Objetivo

Traer datos que viven detrás de una API, cargarlos a pandas, graficarlos y guardarlos. Se usa una API pública sin credenciales para que el ejercicio se pueda repetir sin registro.

Datos de entrada

API pública de CoinGecko (no requiere llave)

Qué debes tener presente

- Cada API es distinta: cambian los parámetros, las credenciales y los límites de uso.
- Conviene indicar la moneda y el número de resultados en el propio prompt.
- Si tu red bloquea la API, sustitúyela por cualquier otra API pública sin llave.

Columnas: Respuesta del endpoint de mercados: identificador, símbolo, nombre, precio actual, capitalización de mercado, volumen y variación en 24 h

Entregables: Script `consulta_cripto.py` • `datos_cripto.csv` y la gráfica en PNG • Manejo explícito de errores HTTP



Ejercicio 5 · el prompt

Prompt para Claude Code

Usa la API pública de CoinGecko para traer las 20 criptomonedas más grandes por capitalización de mercado, en dólares. Con esos datos:

1. Cárgalos en un DataFrame de pandas.
2. Crea una gráfica de barras de la capitalización de mercado.
3. Guarda los datos completos como `datos_cripto.csv`.

Muéstrame los parámetros que enviaste en la petición y maneja los errores de red y los límites de uso de la API.



Ejercicio 6 · Web scraping

Objetivo

Obtener datos que no tienen API. El costo real no es escribir el scraper una vez, sino depurarlo; aquí se usa un sitio sencillo para concentrarse en el flujo completo.

Datos de entrada

Wikipedia: «List of largest companies by revenue»

Qué debes tener presente

- Wikipedia es fácil de extraer; sitios con Cloudflare pueden tomar varias iteraciones.
- Los números vienen con símbolos de moneda, comas y notas al pie.
- Es fácil que el modelo devuelva menos filas de las que pediste sin avisarlo.

Columnas: Tabla principal: posición, empresa, industria, ingresos, utilidad, empleados, país

Entregables: Script `scraper_empresas.py` • `empresas_por_ingreso.csv` y la gráfica • Agrupaciones por país e industria



Ejercicio 6 · el prompt

Prompt para Claude Code

Extrae los datos de la página de Wikipedia «List of largest companies by revenue». Toma la tabla principal con nombre de la empresa, posición, ingresos, utilidad, número de empleados, industria y país. Después:

1. Carga los datos en un DataFrame de pandas.
2. Muéstrame las 10 empresas con mayor ingreso.
3. Crea una gráfica de barras horizontal con las 15 principales.
4. Agrupa por país y por industria.
5. Guarda los datos como `empresas_por_ingreso.csv`.

Maneja los problemas de formato en los números (comas, símbolos de moneda y notas al pie).

Punto de atención. Cuenta las filas que te devuelve: si pediste 10 y te muestra 5, el resultado está incompleto aunque el resumen diga «listo». Verificar la salida es parte del ejercicio.



Ejercicio 7 · Ingeniería de características

Objetivo

Usar al modelo como interlocutor para razonar sobre el problema de negocio: qué señales ocultas hay en los datos y qué indica que un cliente está en riesgo. No sustituye el conocimiento del dominio, lo acelera.

Datos de entrada

`churn_clientes_crudo.csv` — 100 filas

× 11 columnas

Qué debes tener presente

- «Churn» es el abandono o fuga de clientes; la columna objetivo es abandono.
- Cerca de la mitad de los clientes del archivo abandonaron.
- `tipo_contrato` separa perfectamente a los dos grupos: eso es una señal de alarma.

Columnas: `id_cliente`, `edad`, `meses_antiguedad`, `gasto_mensual`, `num_productos`, `tickets_soporte`, `dias_ultimo_acceso`, `tipo_contrato`, `metodo_pago`, `region`, `abandono`

Entregables: `churn_clientes_features.csv` con al menos 8 variables nuevas • `Script features_churn.py` • Tabla de correlaciones con la justificación de cada variable



Ejercicio 7 · el prompt

Prompt para Claude Code

Tengo un conjunto de datos de abandono de clientes en `churn_clientes_crudo.csv` y quiero predecir si un cliente se dará de baja (columna `abandono`). Antes de construir el modelo quiero hacer ingeniería de características:

1. Analiza las variables existentes y sus relaciones.
2. Propón y construye al menos 8 variables nuevas que puedan mejorar la predicción: piensa en razones, tasas, banderas de comportamiento y agrupamientos.
3. Para cada variable nueva, explica por qué crees que será predictiva.
4. Muestra la correlación de cada variable con la variable objetivo.
5. Guarda el conjunto enriquecido como `churn_clientes_features.csv`.

Si alguna variable parece demasiado buena, adviérteme sobre posible fuga de información. Después de tu respuesta te pediré tres variables más, así que deja el script preparado para ampliarse.

Punto de atención. Al final pide pide las otras 3 variables adicionales. Una variable perfectamente predictiva casi siempre es fuga de información.



Ejercicio 8 · Prueba A/B y análisis estadístico

Objetivo

Una prueba A/B parece simple y se puede equivocar de muchas maneras. Aquí se le pide al modelo que verifique el montaje, revise supuestos, elija la prueba adecuada y traduzca el resultado a una recomendación de negocio.

Datos de entrada

`ab_test_landing.csv` — 500 filas $\times$ 8 columnas

Columnas: `id_usuario`, `grupo`, `convirtio`, `ingreso`, `duracion_sesion`, `paginas_vistas`, `dispositivo`, `dia_registro`

Entregables: Script `analisis_ab.py` • Resultados por grupo y por dispositivo • Resumen ejecutivo con recomendación y advertencias

Qué debes tener presente

- 250 usuarios en control (página original) y 250 en tratamiento (diseño nuevo).
- La distribución por dispositivo no está balanceada entre los grupos.
- El ingreso no se distribuye normalmente: la prueba t simple no basta.



Ejercicio 8 · el prompt

Prompt para Claude Code

Ejecuté una prueba A/B en un sitio web y los datos están en `ab_test_landing.csv`. El grupo control vio la página original y el grupo tratamiento vio el diseño nuevo. Necesito que:

1. Verifique que la prueba esté bien montada: balance de los grupos y distribución por dispositivo.
2. Revise los supuestos estadísticos de cada prueba que uses.
3. Calcule el efecto mínimo detectable y la potencia del experimento.
4. Corras la prueba estadística apropiada para la tasa de conversión.
5. Analice el ingreso por usuario con la prueba adecuada para datos no normales.
6. Segmentes los resultados por tipo de dispositivo.

Termina con un resumen ejecutivo en lenguaje sencillo: ¿el diseño nuevo funciona y deberíamos lanzarlo? Señala cualquier riesgo o sesgo que encuentres.



Ejercicio 9 · Pipeline de machine learning de punta a punta

Objetivo

Escribir de una sola vez el proceso para realizar un sistema que prediga el abandono de clientes: partición estratificada, imputación, codificación, escalamiento, validación cruzada y búsqueda de hiperparámetros.

Datos de entrada

`churn_clientes_features.csv` — salida del ejercicio 7

Columnas: Variables originales más las derivadas: `gasto_por_producto`, `tasa_tickets_soporte`, `puntaje_interaccion`, `bandera_alto_riesgo`, `gasto_anual_estimado`, `grupo_antiguedad`, `contrato_mensual`

Entregables: Script `pipeline_churn.py` • `modelo_churn.pkl` y la lista de variables de entrada • Comparación de modelos y gráfica de importancia de variables

Qué debes tener presente

- El objetivo es abandono; el conjunto es pequeño y muy limpio a propósito.
- Con datos así es normal alcanzar exactitud cercana a 100 %: eso no es un buen resultado.
- El modelo entrenado se guarda en disco porque el ejercicio 10 lo necesita.



Ejercicio 9 · el prompt

Prompt para Claude Code

Tengo un conjunto de datos de abandono de clientes en `churn_clientes_features.csv`. La variable objetivo es `abandono`. Construye un pipeline de machine learning completo:

1. Divide en entrenamiento y prueba 80/20 de forma estratificada.
2. Imputa los valores faltantes y codifica las variables categóricas.
3. Escala las variables numéricas.
4. Entrena y compara regresión logística, random forest y gradient boosting.
5. Reporta exactitud, precisión, recall y F1 con validación cruzada.
6. Ajusta los hiperparámetros del mejor modelo con búsqueda en malla.
7. Muestra la importancia de las variables.
8. Guarda el mejor modelo como `modelo_churn.pkl` junto con la lista de variables usadas.

Dime qué modelo funciona mejor y por qué, y adviérteme si las métricas son sospechosamente altas.

Punto de atención. Este es el único ejercicio cuyo entregable no cabe en un archivo de Excel: `modelo_churn.pkl` es un objeto serializado de Python y es la entrada del ejercicio 10.



Ejercicio 10 · Aplicación de Streamlit a partir del modelo

Objetivo

Cerrar el ciclo: un modelo entrenado no sirve de nada guardado en disco. Se construye una interfaz sencilla que cualquier persona del negocio pueda usar.

Datos de entrada

`modelo_churn.pkl` — generado en el ejercicio 9

Qué debes tener presente

- Streamlit se ejecuta localmente con `streamlit run` y también se puede desplegar en la nube.
- Si falta la biblioteca, Claude Code la instala y vuelve a intentar.
- La app hereda las limitaciones del modelo: con 100 filas de entrenamiento no es confiable.

Columnas: Las mismas variables con las que se entrenó el modelo

Entregables: `app_prediccion_churn.py` funcionando en local • Captura de pantalla de la app con una predicción • Nota sobre las limitaciones del modelo



Ejercicio 10 · el prompt

Prompt para Claude Code

Tengo un modelo de abandono de clientes ya entrenado y guardado en `modelo_churn.pkl`, con la lista de variables en el mismo directorio. Construye una aplicación de Streamlit llamada `app_prediccion_churn.py` que tenga:

1. Un título y una descripción claros.
2. Un campo de entrada por cada variable del modelo, con rangos razonables.
3. Un botón que ejecute la predicción.
4. La probabilidad de abandono como porcentaje, con un semáforo de riesgo.
5. Una explicación de las variables que más pesaron en esa predicción.
6. Una barra lateral con instrucciones de uso.
7. Un diseño limpio y profesional, fácil de usar para alguien no técnico.

Ejecútala localmente, instala lo que falte y dime en qué dirección queda escuchando.



Lista de verificación antes de entregar

Sobre el resultado

- ¿Devolvió todo lo que pediste, en la cantidad que pediste?
- ¿Los totales cuadran con el archivo original?
- ¿Detectó los nulos, los atípicos y los valores imposibles?
- ¿Declaró sus supuestos en lugar de inventar datos?

Sobre el proceso

- ¿Te quedaste con un script reejecutable?
- ¿El código está comentado y lo entiendes?
- ¿Una métrica perfecta esconde fuga de información?
- ¿Podrías defender el resultado sin el modelo enfrente?





¿Preguntas?

10 casos de uso reales · ejercicios guiados