



UNIVERSIDAD  
DE MÁLAGA

FACULTAD DE CIENCIAS  
ECONÓMICAS Y EMPRESARIALES

---

# Introducción al Machine Learning con Python

---

Programa Formativo

Duración: 6 horas (3 sesiones de 2 h) · Modalidad: presencial / online

**Nivel:** Introductorio — Sin conocimientos previos requeridos

**Formato:** Teórico-práctico con Jupyter Notebooks

**Entorno:** Google Colab y Kaggle Notebooks (sin instalación local)

**Profesor:** Gonzalo Llamosas García

Curso académico 2025–2026

# Índice

---

<b>1. Información general del curso</b>	<b>3</b>
1.1. Descripción . . . . .	3
1.2. Datos del curso . . . . .	3
1.3. Objetivos generales . . . . .	3
1.4. Metodología . . . . .	4
1.5. Requisitos previos . . . . .	4
1.6. Distribución temporal . . . . .	4
<b>2. Bloque 1 — Fundamentos (Sesión 1: 2 horas)</b>	<b>5</b>
2.1. Notebook 1.1 — Introducción y entorno (~40 min) . . . . .	5
2.1.1. ¿Qué es el Machine Learning? . . . . .	5
2.1.2. Tour por los entornos de trabajo . . . . .	5
2.1.3. Python esencial (repasso rápido) . . . . .	6
2.2. Notebook 1.2 — Python para datos (~80 min) . . . . .	6
2.2.1. NumPy: computación numérica . . . . .	6
2.2.2. Pandas: manipulación de datos tabulares . . . . .	6
2.2.3. Visualización con Matplotlib y Seaborn . . . . .	7
<b>3. Bloque 2 — Aprendizaje supervisado: Regresión (Sesión 2, primera hora)</b>	<b>8</b>
3.1. Notebook 2.1 — Regresión lineal (~30 min) . . . . .	8
3.1.1. Concepto intuitivo . . . . .	8
3.1.2. Flujo de trabajo en ML . . . . .	8
3.1.3. Implementación con scikit-learn . . . . .	8
3.1.4. Métricas de evaluación . . . . .	9
3.2. Notebook 2.2 — Ejercicio guiado: predicción de precios de vivienda (~30 min) . . . . .	9
3.2.1. Dataset . . . . .	9
3.2.2. Desarrollo paso a paso . . . . .	9
<b>4. Bloque 3 — Aprendizaje supervisado: Clasificación (Sesión 2, segunda hora)</b>	<b>10</b>
4.1. Notebook 3.1 — Clasificación (~30 min) . . . . .	10
4.1.1. Concepto . . . . .	10
4.1.2. Regresión logística . . . . .	10
4.1.3. Árboles de decisión . . . . .	10
4.1.4. Métricas de clasificación . . . . .	11
4.2. Notebook 3.2 — Ejercicio guiado: predicción de abandono de clientes (~30 min) . . . . .	11
4.2.1. Dataset . . . . .	11
4.2.2. Desarrollo paso a paso . . . . .	11
<b>5. Bloque 4 — Más allá de lo básico (Sesión 3: 2 horas)</b>	<b>12</b>
5.1. Notebook 4.1 — Aprendizaje no supervisado: Clustering (~35 min) . . . . .	12
5.1.1. Concepto . . . . .	12
5.1.2. K-Means . . . . .	12
5.2. Notebook 4.2 — Introducción a redes neuronales (~50 min) . . . . .	12
5.2.1. Concepto intuitivo . . . . .	12
5.2.2. Implementación con Keras/TensorFlow . . . . .	13
5.3. Notebook 4.3 — Cierre y próximos pasos (~35 min) . . . . .	13
5.3.1. Overfitting vs. Underfitting . . . . .	13
5.3.2. Ética y sesgo en Machine Learning . . . . .	14
5.3.3. Resumen: el flujo completo de un proyecto ML . . . . .	14

5.3.4. Mapa del Machine Learning . . . . .	14
5.3.5. Recursos para seguir aprendiendo . . . . .	14
<b>6. Material entregable y recursos</b>	<b>15</b>
6.1. Notebooks incluidos . . . . .	15
6.2. Datasets utilizados . . . . .	15
6.3. Bibliotecas Python utilizadas . . . . .	15

## Información general del curso

---

### Descripción

Este curso ofrece una introducción práctica y accesible al **Machine Learning** (aprendizaje automático) utilizando el lenguaje de programación **Python**. Está diseñado para personas sin experiencia previa en programación ni en ciencia de datos, y proporciona los fundamentos teóricos y prácticos necesarios para comprender y aplicar las técnicas básicas de ML.

A lo largo de las 6 horas de formación, los alumnos trabajarán con **Jupyter Notebooks** en dos entornos en la nube: **Google Colab** y **Kaggle Notebooks**. Ambas plataformas no requieren instalación local, lo que permite centrarse desde el primer momento en el aprendizaje de los conceptos y herramientas. Kaggle, además, ofrece acceso directo a miles de datasets públicos y una comunidad activa de ciencia de datos.

### Datos del curso

- **Centro:** Facultad de Ciencias Económicas y Empresariales, Universidad de Málaga.
- **Profesor:** Gonzalo Llamosas García.
- **Duración:** 6 horas (3 sesiones de 2 horas).
- **Dirigido a:** Alumnado de la Facultad de Ciencias Económicas y Empresariales.

### Objetivos generales

Al finalizar el curso, los alumnos serán capaces de:

1. Comprender qué es el Machine Learning y distinguir sus principales paradigmas (supervisado, no supervisado y por refuerzo).
2. Manejar las herramientas fundamentales del ecosistema Python para datos: NumPy, Pandas, Matplotlib y Seaborn.
3. Implementar modelos de **regresión** y **clasificación** con scikit-learn siguiendo un flujo de trabajo estructurado.
4. Aplicar técnicas de **clustering** (aprendizaje no supervisado) con K-Means.
5. Construir y entrenar una **red neuronal básica** con Keras/TensorFlow.
6. Evaluar modelos mediante métricas apropiadas e interpretar los resultados.
7. Conocer los conceptos de overfitting, underfitting y validación cruzada.
8. Reflexionar sobre las implicaciones éticas del Machine Learning en el ámbito económico y empresarial.

## Metodología

[title=Enfoque metodológico] La formación sigue un enfoque **teórico-práctico** estructurado en ciclos de:

1. **Explicación conceptual** — Presentación del concepto con ejemplos intuitivos y visuales.
2. **Demostración con código** — El profesor ejecuta y explica el código en vivo.
3. **Práctica guiada** — Los alumnos replican y modifican el código en sus propios notebooks.
4. **Ejercicio autónomo** — Pequeños retos para consolidar lo aprendido.

Cada bloque incluye un **Jupyter Notebook autocontenido** con explicaciones en Markdown, código ejecutable y ejercicios. Los alumnos conservan todos los notebooks al finalizar.

## Requisitos previos

- **Conocimientos:** No se requiere experiencia en programación ni en estadística. Se proporcionarán las bases necesarias durante la formación.
- **Equipamiento:** Ordenador portátil con navegador web moderno (Chrome, Firefox o Edge).
- **Cuentas:** Cuenta de Google (para Google Colab) y cuenta de Kaggle (gratuita, en <https://www.kaggle.com>).

## Distribución temporal

El curso se distribuye en **3 sesiones de 2 horas** cada una, lo que permite asentar lo aprendido entre sesiones.

Sesión	Bloques	Contenido principal	Duración
1	Bloque 1	Introducción al ML, Python para datos (NumPy, Pandas, visualización)	2 h
2	Bloques 2-3	Regresión lineal y clasificación (regresión logística, árboles de decisión)	2 h
3	Bloque 4	Clustering, redes neuronales, overfitting, ética y sesgo, cierre	2 h
<b>Total</b>			<b>6 h</b>

## Bloque 1 — Fundamentos (Sesión 1: 2 horas)

### Objetivos de aprendizaje

- Comprender qué es el Machine Learning y sus tipos principales.
- Familiarizarse con los entornos Google Colab y Kaggle Notebooks.
- Dominar las operaciones básicas con NumPy y Pandas.
- Crear visualizaciones básicas de datos.

### Notebook 1.1 — Introducción y entorno (~40 min)

#### ¿Qué es el Machine Learning?

- **Definición accesible:** sistemas que aprenden patrones a partir de datos, en lugar de ser programados explícitamente con reglas.
- **Diferencia con la programación tradicional:**
  - Programación clásica: *datos + reglas* → *respuesta*
  - Machine Learning: *datos + respuestas* → *reglas (modelo)*
- **Tipos de aprendizaje:**
  - **Supervisado:** se proporcionan datos etiquetados (entrada-salida). Ej.: predecir las ventas de un producto a partir de la inversión en publicidad.
  - **No supervisado:** no hay etiquetas; el modelo descubre patrones ocultos. Ej.: segmentar clientes por comportamiento de compra.
  - **Por refuerzo:** un agente aprende mediante ensayo y error, recibiendo recompensas. Ej.: optimización dinámica de precios en tiempo real.
- **Ejemplos en el ámbito económico y empresarial:** detección de fraude bancario, *scoring* crediticio, predicción de abandono de clientes (*churn*), segmentación de mercado, previsión de demanda, análisis de riesgo financiero.
- **Otros ejemplos cotidianos:** filtro de spam, recomendaciones de Netflix/Spotify, asistentes de voz.

#### Tour por los entornos de trabajo

##### Google Colab:

- Acceso: <https://colab.research.google.com>
- Estructura de un notebook: celdas de código y celdas de texto (Markdown).
- Ejecución de celdas: **Shift + Enter**.
- Cómo guardar, compartir y descargar notebooks.
- Breve mención al uso de GPU (menú *Entorno de ejecución*).

##### Kaggle Notebooks:

- Acceso: <https://www.kaggle.com/code>

- Diferencias con Colab: interfaz, acceso a datasets de Kaggle, sistema de versiones.
- Cómo crear un notebook, añadir datasets y ejecutar código.
- Ventaja: acceso directo a datasets públicos sin necesidad de descargar archivos.
- GPU disponible de forma gratuita (menú *Settings* → *Accelerator*).

*Nota: Ambos entornos son equivalentes para este curso. Los notebooks proporcionados funcionan en cualquiera de los dos sin modificaciones.*

### Python esencial (repaso rápido)

- Variables, tipos de datos básicos (`int`, `float`, `str`, `list`).
- Operaciones aritméticas y lógicas.
- Estructuras de control: `if/else`, `for`.
- Funciones: definición con `def`, parámetros, `return`.
- Importar bibliotecas: `import numpy as np`.

### Notebook 1.2 — Python para datos (~80 min)

#### NumPy: computación numérica

- ¿Por qué NumPy? Velocidad y eficiencia frente a listas de Python.
- Crear arrays: `np.array()`, `np.zeros()`, `np.ones()`, `np.linspace()`.
- Operaciones vectorizadas: suma, resta, multiplicación elemento a elemento.
- Indexación y slicing de arrays.
- Funciones útiles: `np.mean()`, `np.std()`, `np.max()`, `np.reshape()`.

#### Pandas: manipulación de datos tabulares

- Concepto de `DataFrame` y `Series`.
- Carga de datos: `pd.read_csv()` con un dataset de ejemplo.
- Exploración básica:
  - `df.head()`, `df.tail()`, `df.shape`
  - `df.info()`, `df.describe()`
  - `df.columns`, `df.dtypes`
- Selección de columnas y filas: `df['col']`, `df.loc[]`, `df.iloc[]`.
- Filtrado de datos: `df[df['col'] > valor]`.
- Gestión de valores nulos: `df.isnull().sum()`, `df.dropna()`, `df.fillna()`.

## Visualización con Matplotlib y Seaborn

- Filosofía: “una imagen vale más que mil números”.
- **Matplotlib** — gráficos básicos:
  - Gráfico de líneas: `plt.plot()`
  - Gráfico de dispersión: `plt.scatter()`
  - Histogramas: `plt.hist()`
  - Personalización: títulos, ejes, leyendas, colores.
- **Seaborn** — gráficos estadísticos de alto nivel:
  - `sns.heatmap()` para matrices de correlación.
  - `sns.pairplot()` para explorar relaciones entre variables.
  - `sns.boxplot()` para distribuciones.

## Bloque 2 — Aprendizaje supervisado: Regresión (Sesión 2, primera hora)

### Objetivos de aprendizaje

- Comprender el concepto de regresión y cuándo aplicarla.
- Implementar el flujo completo de un proyecto de ML: datos → modelo → evaluación.
- Entender la importancia de separar datos en entrenamiento y test.
- Interpretar métricas de regresión (MSE, RMSE,  $R^2$ ).

### Notebook 2.1 — Regresión lineal (~30 min)

#### Concepto intuitivo

- **Problema de regresión:** predecir un valor numérico continuo.
- Ejemplos del ámbito económico: predecir ventas mensuales, estimar el precio de un inmueble, prever la demanda de un producto, proyectar ingresos futuros.
- **Regresión lineal simple:** encontrar la recta que mejor se ajusta a los datos ( $y = mx + b$ ).
- **Regresión lineal múltiple:** múltiples variables de entrada ( $y = w_1x_1 + w_2x_2 + \dots + b$ ).
- Visualización: gráfico de dispersión con la línea de regresión superpuesta.

#### Flujo de trabajo en ML

Se presenta el flujo de trabajo estándar que se repetirá en todos los bloques:

1. **Cargar y explorar** los datos.
2. **Preprocesar:** limpiar, seleccionar variables relevantes.
3. **Separar** en conjunto de entrenamiento (*train*) y de prueba (*test*).
4. **Entrenar** el modelo con los datos de entrenamiento.
5. **Predecir** sobre los datos de prueba.
6. **Evaluar** el rendimiento del modelo.

#### Implementación con scikit-learn

- Introducción a **scikit-learn**: la biblioteca estándar de ML en Python.
- API consistente: `model.fit()`, `model.predict()`, `model.score()`.
- `train_test_split()`: separación de datos (concepto de *random\_state* para reproducibilidad).
- `LinearRegression()`: creación, ajuste y predicción.
- Visualizar predicciones vs. valores reales.

## Métricas de evaluación

- **MSE** (Error Cuadrático Medio): penaliza errores grandes.
- **RMSE** (Raíz del MSE): interpretable en las mismas unidades que la variable objetivo.
- **R<sup>2</sup>** (Coeficiente de determinación): proporción de varianza explicada por el modelo (0 = malo, 1 = perfecto).
- Cómo interpretar estas métricas en lenguaje no técnico.

## Notebook 2.2 — Ejercicio guiado: predicción de precios de vivienda (~30 min)

### Dataset

- Se utilizará el dataset de **California Housing** (incluido en scikit-learn), un caso de estudio clásico en economía inmobiliaria.
- Variables: ingreso medio de la zona, antigüedad de la vivienda, número de habitaciones, población, ubicación geográfica, etc.
- Variable objetivo: valor medio de la vivienda.
- **Contexto económico:** ¿qué factores determinan el precio de la vivienda? ¿Cómo influye la renta de la zona? Conexión con modelos econométricos que los alumnos puedan conocer.

### Desarrollo paso a paso

1. Carga del dataset con `sklearn.datasets.fetch_california_housing()`.
2. Análisis exploratorio: distribuciones, correlaciones, scatter plots.
3. Selección de variables predictoras.
4. División train/test (80%/20%).
5. Entrenamiento de un modelo `LinearRegression`.
6. Evaluación: cálculo de MSE, RMSE y R<sup>2</sup>.
7. Visualización: predicciones vs. reales, gráfico de residuos.
8. **Experimentación:** ¿qué pasa si usamos menos variables? ¿Y si cambiamos el tamaño del test set?

## Bloque 3 — Aprendizaje supervisado: Clasificación (Sesión 2, segunda hora)

### Objetivos de aprendizaje

- Comprender la diferencia entre regresión y clasificación.
- Implementar modelos de clasificación: regresión logística y árboles de decisión.
- Interpretar métricas de clasificación: accuracy, precisión, recall, matriz de confusión.
- Comparar el rendimiento de dos modelos sobre el mismo dataset.

### Notebook 3.1 — Clasificación (~30 min)

#### Concepto

- **Problema de clasificación:** predecir una categoría o clase discreta.
- Ejemplos del ámbito económico: aprobación de crédito (sí/no), detección de fraude en transacciones, predicción de abandono de clientes (*churn*), clasificación de riesgo financiero (bajo/medio/alto).
- Clasificación binaria vs. multiclase.

#### Regresión logística

- A pesar de su nombre, es un algoritmo de **clasificación**.
- Concepto: aplica una función sigmoide para obtener probabilidades (0 a 1).
- Umbral de decisión: si  $P > 0,5 \rightarrow$  clase 1, si no  $\rightarrow$  clase 0.
- Implementación con `LogisticRegression()` de scikit-learn.
- Visualización de la frontera de decisión (en 2D).

#### Árboles de decisión

- Concepto intuitivo: una serie de preguntas de sí/no que van segmentando los datos.
- Analogía: un diagrama de flujo para tomar decisiones.
- Ventaja: fácilmente interpretables y visualizables.
- Implementación con `DecisionTreeClassifier()`.
- Visualización del árbol con `plot_tree()`.
- Riesgo de **sobreajuste** si el árbol es muy profundo (anticipo del concepto de overfitting del Bloque 4).

## Métricas de clasificación

- **Accuracy:** proporción de predicciones correctas. Limitaciones cuando las clases están desbalanceadas.
- **Matriz de confusión:** tabla de verdaderos/falsos positivos/negativos. Lectura e interpretación.
- **Precisión** (*precision*): de los que predije positivos, ¿cuántos lo eran realmente?
- **Recall** (*sensibilidad*): de los positivos reales, ¿cuántos detecté?
- **F1-score:** media armónica de precisión y recall.
- `classification_report()` de scikit-learn como herramienta resumen.

## Notebook 3.2 — Ejercicio guiado: predicción de abandono de clientes (~30 min)

### Dataset

- Dataset de **Telco Customer Churn** (disponible en Kaggle): predecir si un cliente abandonará el servicio.
- Variables: antigüedad del cliente, tipo de contrato, método de pago, gasto mensual, servicios contratados, etc.
- Variable objetivo: **Churn** (0 = se queda, 1 = abandona).
- **Contexto empresarial:** la retención de clientes es un problema clave en marketing y gestión. Predecir qué clientes tienen riesgo de irse permite tomar medidas preventivas.

### Desarrollo paso a paso

1. Carga y exploración del dataset.
2. **Preprocesamiento:**
  - Gestión de valores nulos.
  - Codificación de variables categóricas (`pd.get_dummies()` o `LabelEncoder`).
  - Selección de variables relevantes.
3. División train/test.
4. Entrenamiento de **Regresión Logística**.
5. Entrenamiento de **Árbol de Decisión**.
6. Evaluación comparativa: accuracy, matrices de confusión, `classification report`.
7. **Discusión:** ¿qué modelo funciona mejor? ¿Qué variables influyen más en la fuga de clientes? ¿Cómo usaría una empresa esta información?

## Bloque 4 — Más allá de lo básico (Sesión 3: 2 horas)

### Objetivos de aprendizaje

- Comprender y aplicar K-Means para agrupar datos sin etiquetas.
- Construir y entrenar una red neuronal sencilla con Keras.
- Entender los conceptos de overfitting y underfitting.
- Reflexionar sobre la ética, el sesgo y la responsabilidad en el uso de modelos de ML.
- Conocer el mapa general del Machine Learning y caminos para seguir aprendiendo.

### Notebook 4.1 — Aprendizaje no supervisado: Clustering (~35 min)

#### Concepto

- **Aprendizaje no supervisado:** no hay etiquetas; el modelo busca estructura oculta en los datos.
- **Clustering:** agrupar datos similares en *clusters* (grupos).
- Aplicaciones en el ámbito empresarial: segmentación de clientes por perfil de consumo, agrupación de productos similares, detección de transacciones anómalas, segmentación de mercados.

#### K-Means

- Algoritmo intuitivo:
  1. Elegir  $K$  (número de clusters).
  2. Asignar cada punto al centroide más cercano.
  3. Recalcular centroides.
  4. Repetir hasta convergencia.
- Implementación con `KMeans()` de scikit-learn.
- Visualización de los clusters en 2D (scatter plot coloreado por cluster).
- ¿Cómo elegir  $K$ ? Método del codo (*elbow method*): gráfico de inercia vs.  $K$ .

### Notebook 4.2 — Introducción a redes neuronales (~50 min)

#### Concepto intuitivo

- **Inspiración biológica:** neuronas artificiales inspiradas (de forma simplificada) en el cerebro.
- **Neurona artificial:** recibe entradas, aplica pesos y un sesgo (*bias*), pasa el resultado por una función de activación.
- **Capas:**
  - *Input layer*: recibe los datos de entrada.
  - *Hidden layers*: capas intermedias que extraen características.

- *Output layer*: produce la predicción final.
- **Funciones de activación**: ReLU (capas ocultas), Softmax/Sigmoid (capa de salida).
- **Entrenamiento**: el modelo ajusta los pesos para minimizar una función de pérdida (*loss*) mediante **descenso del gradiente** (explicación visual, sin entrar en derivadas).
- Diagrama visual de una red de 2–3 capas.

### Implementación con Keras/TensorFlow

- **Keras**: API de alto nivel para construir redes neuronales. Viene integrada en TensorFlow (preinstalado en Colab y Kaggle).
- Dataset: **MNIST** (dígitos escritos a mano, 28×28 píxeles, 10 clases).
- **Construcción del modelo** (~10 líneas de código):
  - `Sequential()`: modelo secuencial (capas apiladas).
  - `Dense(128, activation='relu')`: capa oculta de 128 neuronas.
  - `Dense(10, activation='softmax')`: capa de salida (10 clases).
- **Compilación**: elegir optimizador (`adam`), función de pérdida (`sparse_categorical_crossentropy`), métrica (`accuracy`).
- **Entrenamiento**: `model.fit(X_train, y_train, epochs=10, validation_split=0.2)`.
- Observar en vivo cómo la *loss* baja y la *accuracy* sube en cada época.
- **Evaluación**: `model.evaluate(X_test, y_test)`.
- **Comparación**: entrenar un clasificador clásico de scikit-learn (e.g., `LogisticRegression`) sobre el mismo dataset MNIST y contrastar su rendimiento con el de la red neuronal.

### Notebook 4.3 — Cierre y próximos pasos (~35 min)

#### Overfitting vs. Underfitting

- **Underfitting**: el modelo es demasiado simple y no captura los patrones de los datos. Alto error en train y en test.
- **Overfitting**: el modelo memoriza los datos de entrenamiento y no generaliza. Bajo error en train, alto error en test.
- Visualización gráfica: curvas de complejidad, comparación train/test error.
- **Validación cruzada** (*cross-validation*): técnica para estimar mejor el rendimiento real del modelo.
- Breve mención a técnicas para combatir el overfitting: regularización, *dropout* (en redes neuronales), poda de árboles, más datos.

## Ética y sesgo en Machine Learning

- **Sesgo algorítmico:** los modelos aprenden de datos históricos, que pueden contener sesgos humanos. Ej.: un modelo de concesión de créditos entrenado con datos históricos puede discriminar por género o etnia.
- **Equidad (*fairness*):** ¿es justo que un algoritmo tome decisiones que afectan a personas? Casos reales en banca, seguros y contratación.
- **Transparencia e interpretabilidad:** importancia de entender *por qué* un modelo toma una decisión, especialmente en contextos regulados (sector financiero, RGPD).
- **Responsabilidad:** ¿quién responde cuando un modelo se equivoca? Breve discusión.

## Resumen: el flujo completo de un proyecto ML

1. **Definir el problema:** ¿qué queremos predecir? ¿Es regresión o clasificación?
2. **Obtener y explorar datos:** Pandas, visualización.
3. **Preprocesar:** limpieza, codificación, normalización.
4. **Seleccionar y entrenar modelos:** probar varios, comparar.
5. **Evaluar:** métricas adecuadas, validación cruzada.
6. **Iterar:** mejorar, ajustar hiperparámetros, probar nuevas variables.
7. **Comunicar resultados.**

## Mapa del Machine Learning

Ubicación de lo aprendido en el contexto más amplio:

- **Lo que hemos visto:** regresión lineal, regresión logística, árboles de decisión, K-Means, red neuronal básica.
- **Siguiente nivel (ML clásico):** Random Forest, Gradient Boosting (XGBoost, LightGBM), SVM, PCA.
- **Deep Learning:** redes convolucionales (imágenes), redes recurrentes — RNN/LSTM — (series temporales, texto; especialmente relevantes en economía para predicción de ventas, demanda o mercados financieros), transformers (GPT, BERT).
- **MLOps:** despliegue de modelos, monitorización, pipelines.

## Recursos para seguir aprendiendo

- **Cursos online:** Andrew Ng — Machine Learning (Coursera), fast.ai, Google ML Crash Course.
- **Libros:** *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* (Aurélien Géron).
- **Plataformas de práctica:** Kaggle (competiciones, datasets y notebooks), Google Colab. Ambas utilizadas durante el curso.
- **Documentación oficial:** [scikit-learn.org](https://scikit-learn.org), [tensorflow.org/tutorials](https://tensorflow.org/tutorials).
- **Comunidad:** Stack Overflow, Reddit (r/MachineLearning, r/learnmachinelearning).

## Material entregable y recursos

### Notebooks incluidos

Nº	Notebook	Contenido
1.1	01_intro_entorno.ipynb	Introducción a ML, tour por Colab y Kaggle, Python básico
1.2	02_python_datos.ipynb	NumPy, Pandas, Matplotlib, Seaborn
2.1	03_regresion_lineal.ipynb	Regresión lineal con scikit-learn
2.2	04_ejercicio_regresion.ipynb	Ejercicio guiado: precios de vivienda
3.1	05_clasificacion.ipynb	Regresión logística y árboles de decisión
3.2	06_ejercicio_clasificacion.ipynb	Ejercicio guiado: abandono de clientes ( <i>churn</i> )
4.1	07_clustering.ipynb	K-Means y método del codo
4.2	08_red_neuronal.ipynb	Red neuronal con Keras/TensorFlow
4.3	09_cierre.ipynb	Overfitting, ética y sesgo, resumen y recursos

### Datasets utilizados

- **California Housing** (scikit-learn): regresión, predicción de precios.
- **Telco Customer Churn** (Kaggle): clasificación binaria, predicción de abandono de clientes.
- **MNIST** (Keras/TensorFlow): clasificación de dígitos con red neuronal.
- **Datos sintéticos** (make\_blobs): clustering con K-Means.

### Bibliotecas Python utilizadas

Biblioteca	Versión mín.	Uso
numpy	1.21+	Computación numérica con arrays
pandas	1.3+	Manipulación de datos tabulares
matplotlib	3.4+	Visualización de datos
seaborn	0.11+	Visualización estadística
scikit-learn	1.0+	Modelos de ML clásico
tensorflow	2.8+	Redes neuronales (incluye Keras)

*Nota: Todas estas bibliotecas vienen preinstaladas tanto en Google Colab como en Kaggle Notebooks, por lo que no es necesaria ninguna instalación adicional.*