Business Analytics & Data Science

Día 6: Introducción a Machine Learning

EAE Business School Barcelona

9 de febrero de 2026

Plan del Día 6

Primera Parte (9:00-11:00)

1. Recap Semana 1
2. ¿Qué es Machine Learning?
3. El workflow de ML

Segunda Parte (11:30-13:30)

1. Conceptos de regresión lineal
2. Regresión lineal con scikit-learn
3. Preview: evaluación de modelos

Evaluación (13:30-14:00)

Recap Semana 1

Python Fundamentals (Días 1-3):

- Variables, control de flujo, funciones
- Listas, diccionarios, pandas
- Limpieza de datos, visualización

Estadística (Días 4-5):

- Descriptiva: media, std, percentiles
- Distribuciones: normal, scipy.stats
- Inferencia: intervalo de confianza, t-tests, chi-cuadrado

Primera Parte: ¿Qué es Machine Learning?

Objetivo

Si la semana pasada intentábamos estudiar qué características influían sobre

- Precio de viviendas
- Riesgo de cancelación de reservas
- ADR de clientes de hotel

En las sesiones restantes directamente intentaremos **predecir** los valores de estas variables a partir de las otras características presentes en el dataset.

En todos los casos, la clave es **estudiar la probabilidad** del fenómeno que nos interesa.

Definición de Machine Learning

Machine Learning == **Aprendizaje Automático**

Machine Learning:

- En lugar de describir reglas explícitas, proporcionamos datos + ejemplos
- Algoritmo aprende las reglas automáticamente
- Hace predicciones sobre datos nuevos

Clave: El modelo **generaliza** a partir de ejemplos

Ejemplos Reales de uso de ML

Reconocimiento de imágenes: Detectar objetos en fotos
Recomendaciones: Netflix, Spotify, Amazon
Detección de fraude: Transacciones bancarias sospechosas
Predicción de precios: Viviendas, acciones, demanda
Procesamiento de lenguaje: Traducción, chatbots
Diagnóstico médico: Detectar enfermedades en imágenes

En común:

Aprender de *datos conocidos* para predecir sobre *datos no vistos previamente*

Un poco de historia

- 1950s-60s: **Fundación**
- 1970s-80s: **AI Winter**
- 1990s-2000s: **Renacimiento estadístico** / Primeras aplicaciones prácticas
  - Gran disponibilidad de datos
- 2010s: **Deep Learning**
  - Aparece el entrenamiento a gran escala con **GPUs**
- 2020s: La Era de los **LLM**
  - La IA se convierte en una tecnología de uso general

Tipos de Machine Learning

1. **Aprendizaje Supervisado** (conmigo)

- Tenemos ejemplos con respuestas correctas (etiquetas)
- El modelo aprende la relación entrada → salida
- Ejemplos: Predecir precio, clasificar spam

2. **Aprendizaje No Supervisado** (con Carlos Barahona)

- Solo tenemos datos, sin etiquetas
- El modelo encuentra patrones/grupos
- Ejemplos: Segmentación de clientes, detección de anomalías

Supervisado: Regresión vs Clasificación

Regresión = predecir valor numérico continuo

- Ejemplos:
  - Predecir precio de vivienda (€)
  - Predecir ADR de un cliente

Clasificación = predecir categoría discreta

- Ejemplos:
  - Cliente cancelará o no su reserva
  - Email es spam o no spam

El Workflow de Machine Learning

```
1. Datos
   ↓
2. Features (características)
   ↓
3. Modelo (algoritmo de ML)
   ↓
4. Predicciones
   ↓
5. Evaluación
```

Iterativo: Si evaluación es mala → ajustar features/modelo y repetir

1. Datos

Necesitamos:

- Datos históricos *limpios* con ejemplos
- Variable objetivo (target): lo que queremos predecir
- Variables predictoras (features): información útil

Ejemplo predicción de precios:

- Target: `price`
- Features: `sqrmts`, `rooms`, `neighborhood`, `floor`, etc.

Más datos (generalmente) = mejor modelo

2. Features

Features = variables que el modelo usa para aprender

Buenas features:

- Correlacionadas con el target
- Información útil, no ruido
- Bien formateadas (números, no texto)

Feature engineering:

- Crear nuevas features: `price_per_sqm = price / sqrmts`
- Transformar categóricas: `neighborhood` → one-hot encoding
- Escalar/normalizar valores

"Features make or break ML models"

3. Modelo

Modelo = algoritmo que aprende de los datos

Elegir modelo depende de:

- Tipo de problema (regresión vs clasificación)
- Cantidad de datos
- Interpretabilidad vs precisión

Hoy: Regresión Lineal

4. Predicciones

Una vez entrenado, el modelo puede predecir sobre **datos nuevos**:

```
# Entrenar modelo con datos históricos
model.fit(X_train, y_train)

# Predecir sobre casos nuevos
y_pred = model.predict(X_test)
```

Importante: Predecir sobre datos que el modelo **nunca ha visto**

5. Evaluación

¿Cómo sabemos si el modelo es bueno?

- Comparar predicciones vs valores reales
- Métricas: MAE, RMSE, R^2 (veremos mañana)
- **Crítico**: Evaluar en datos de TEST, no de entrenamiento

Train/Test Split:

- Datos de entrenamiento: para aprender
- Datos de test: para evaluar (sin trampa)

Train/Test Split

Problema: Si evaluamos en datos de entrenamiento, el modelo puede "memorizar"

Solución: Dividir datos en dos conjuntos

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

Test set = simulación de datos futuros

🔬 Ahora al notebook

Ejercicio Conceptos de ML

Segunda Parte: Regresión Lineal

Regresión Lineal: Intuición

Objetivo: Si x son los metros cuadrados e y el precio, se trata de encontrar una línea recta

$$\hat{y} = \beta_0 + \beta_1 \cdot x$$

que se ajuste *lo mejor posible* a los datos.

- β_0 **(intercept)**: Precio cuando metros = 0 (punto de corte)
- β_1 **(coef)**: Cuánto aumenta el precio por cada metro adicional

Visualmente: La línea que minimiza distancia a todos los puntos

Mínimos Cuadrados Ordinarios (OLS)

Método: Minimizar la suma de errores al cuadrado:

$$\min_{\beta_0,\beta_1} \sum (y - \hat{y})^2$$

Residuo = diferencia entre valor real y predicción

OLS encuentra los β que minimizan residuos

Regresión Lineal Simple en Python

```
from sklearn.linear_model import LinearRegression

# 1. Preparar datos
X = df[['sqrmts']]  # Features (debe ser 2D)
y = df['price']     # Target (puede ser 1D)

# 2. Crear modelo
model = LinearRegression()
```

```python
# 3. Entrenar (fit)
model.fit(X, y)

# 4. Predecir
predicciones = model.predict(X)
```

fit() = aprende los coeficientes β

Interpretar Coeficientes

Ejemplo output:

```
Intercept: 50000
Coeficiente: 3000
```

Interpretación:

- Intercept: Precio base de 50k€
- Coef: Por cada m² adicional, el precio aumenta 3000€

Ecuación: `precio = 50000 + 3000 × metros`

Visualizar el Ajuste

```
import plotly.express as px

df_viz = df.copy()
df_viz['prediccion'] = model.predict(X)

fig = px.scatter(df_viz, x='sqrmts', y='price',
                 title='Regresión Lineal: Precio vs Metros')
fig.show()
```

Añadir Línea de Regresión

```
import plotly.graph_objects as go

fig.add_trace(go.Scatter(x=df_viz['sqrmts'],
                         y=df_viz['prediccion'],
                         mode='lines', name='Predicción'))
fig.show()
```

La línea atraviesa la nube de puntos

Hacer Predicciones

```python
# Predecir precio de un piso de 85m²
nuevo_piso = [[85]]  # Debe ser 2D
precio_predicho = model.predict(nuevo_piso)[0]

print(f"Piso de 85m²: {precio_predicho:.0f}€")

# Predecir múltiples
pisos_nuevos = [[60], [85], [120]]
precios = model.predict(pisos_nuevos)
```

IMPORTANTE: Predecir solo dentro del rango de entrenamiento (60-150m²)

Limitaciones de Regresión Lineal Simple

Asume:

- Relación lineal (línea recta)
- Solo 1 feature (metros cuadrados)

En realidad:

- Relación puede ser no lineal
- El precio depende de MUCHOS factores: ubicación, habitaciones, etc.

Solución (mañana): Regresión lineal múltiple

🔬 Ahora al notebook

Ejercicio Regresión Lineal

Gracias