Clase 054 — Proyecto end-to-end: visión, datos, exploración, preparación

Parte: 1 — Machine Learning Clásico · Fuente: Géron, cap. 2 (California Housing). Duración estimada: 90 min.

Clase 054 — Proyecto end-to-end: visión, datos, exploración, preparación

> *Parte: 1 — Machine Learning Clásico · Fuente: Géron, cap. 2 (California Housing). Duración estimada: 90 min.*

Objetivo

Que el alumno recorra la primera mitad de un proyecto de ML real de punta a punta: framear el problema en términos de negocio, conseguir los datos, hacer un EDA honesto, separar train/test sin contaminarse, y dejar el pipeline de preparación (limpieza, encoding, scaling) listo para entrenar — todo sobre el dataset California Housing del capítulo 2 de Géron.

Resultados de aprendizaje

Al finalizar la clase, el alumno podrá:

1. Framear el problema en términos de negocio: tipo de tarea (regresión/clasificación), métrica, baseline.
2. Hacer un EDA reproducible: describe, info, hist, corr, scatter matrix, mapas geográficos.
3. Separar train/test correctamente con train_test_split estratificado por una variable clave (income bucket).
4. Construir un pipeline con Pipeline + ColumnTransformer que limpie, encode (OneHotEncoder) y escale (StandardScaler) en un solo objeto.
5. Evitar data leakage: todo cálculo (medias, encodings, scalers) se ajusta solo en train y se aplica en test.

Temas

#	Tema	Por qué importa
1	Framing del problema	Sin objetivo claro y métrica, cualquier mo
2	EDA: describe, hist, corr, geo plot	Conocer los datos antes de modelar.
3	Stratified split por income bucket	Test representativo, no muestreo aleatorio
4	Limpieza: NaN, outliers, tipos	El 60% del trabajo real.
5	Encoding categórico (OneHotEncoder, Ordina	Los modelos no comen strings.
6	Scaling (StandardScaler, MinMaxScaler)	Imprescindible para modelos sensibles a es
7	Pipelines + ColumnTransformer	Reproducible, sin leakage, deployable.

Versión profundizada — 2026

El tema moderno que vivía como complemento dentro de esta clase ahora tiene clase propia dedicada con patrón completo, ejercicios y homework:

- Clase 050b — Feature Engineering avanzado + MICE imputation

Definiciones y características

Pipeline (sklearn)

: Secuencia de transformaciones (nombre, estimator) que termina opcionalmente en un modelo. fit ajusta cada paso con la salida del anterior; transform/predict aplica en orden. Encadena todo en un objeto y previene leakage cuando se usa con CV.

ColumnTransformer

: Aplica transformadores distintos a subconjuntos de columnas (numéricas → scaler, categóricas → encoder) y concatena el resultado. Es el pegamento entre EDA y modelo.

Stratified split

: Separación train/test que preserva la distribución de una variable clave (target en clasificación, o un bucket de una numérica como income_cat). Evita que el test caiga sesgado por azar — crítico en N chico.

OneHotEncoder

: Convierte cada valor único de una categórica en una columna binaria. Default sklearn: sparse matrix. Parámetros clave: handle_unknown='ignore' (para categorías nuevas en test), drop='first' (para evitar multicolinealidad en modelos lineales).

StandardScaler

: Resta media y divide por desvío estándar (z-score). Necesario para modelos sensibles a escala: SVM, KNN, redes neuronales, regresión regularizada (Ridge/Lasso). Árboles no lo necesitan.

Target leakage

: Cualquier feature, encoding o estadístico que contenga información del target o del test al momento de entrenar. Inflará métricas en validación y colapsará en producción. Síntoma típico: AUC > 0.99 sospechoso.

SimpleImputer

: Imputación univariada (media/mediana/moda/constante). Default razonable para empezar; reemplazable por KNNImputer o IterativeImputer sin cambiar el resto del pipeline.

Métrica vs función de costo

: La métrica la elige el negocio (RMSE en dólares, recall en fraude); la función de costo la usa el optimizador internamente (puede ser distinta — MSE para entrenar, RMSE para reportar).

Dataset / recursos

California Housing (Géron cap. 2). 20 640 filas, 10 columnas, target = median_house_value. Disponible vía sklearn.datasets.fetch_california_housing() o el CSV del repo de Géron. Contiene una categórica (ocean_proximity) y una columna con NaN (total_bedrooms) — perfecta para practicar pipeline completo.

Ejercicios

1. EDA mínimo. Cargá el dataset y producí: df.info(), df.describe(), df.hist(bins=50, figsize=(12,8)). Identificá al menos 2 anomalías (cap visual de median_house_value, distribución skewed de population).

2. Stratified split por income bucket. Creá income_cat = pd.cut(df['median_income'], bins=[0, 1.5, 3, 4.5, 6, np.inf]) y usá StratifiedShuffleSplit para train/test 80/20. Verificá que la distribución de income_cat sea casi idéntica en train y test.

3. Target encoding sin leakage. Tomá una variable categórica (creá zipcode_fake a partir de buckets de lat/long si querés). Implementá target encoding con category_encoders.TargetEncoder ajustado solo en train. Compará RMSE de un RandomForestRegressor con (a) one-hot vs (b) target encoded.

4. KNNImputer vs SimpleImputer. Sobre total_bedrooms (que tiene NaN reales), comparen RMSE final del pipeline usando SimpleImputer(strategy='median') vs KNNImputer(n_neighbors=5). Reportá cuál ganó y en cuánto.

5. Pipeline completo. Armá un ColumnTransformer con: numéricas → SimpleImputer(median) + StandardScaler; categóricas → OneHotEncoder(handle_unknown='ignore'). Envolvelo en un Pipeline con LinearRegression al final. fit en train, RMSE en test.

Homework verificable

Notebook con: (a) carga del dataset + EDA con al menos 4 gráficos; (b) stratified split por income bucket con verificación de proporciones; (c) pipeline ColumnTransformer (numéricas con KNNImputer+StandardScaler, categóricas con OneHotEncoder); (d) feature engineering manual con al menos 2 features derivadas (rooms_per_household, bedrooms_per_room); (e) baseline LinearRegression con RMSE reportado en train y test.

Criterio de aceptación: El pipeline se entrena con pipeline.fit(X_train, y_train) sin tocar X_test antes del scoring final. RMSE de test reportado. Sin warnings de sklearn por leakage o categorías nuevas.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
Target encoding da AUC casi perfecto en tr	Encoding calculado sobre todo el dataset →
ValueError: Found unknown categories ['X']	El test trae una categoría nueva no vista
RMSE de test ridículamente bajo, idéntico	Imputaste/escalaste sobre pd.concat([train
fillna(df.mean()) baja drásticamente la va	Imputación univariada con media aplasta la
Modelo lineal con coeficientes raros (uno	Olvidaste el StandardScaler — features est

Preguntas frecuentes

¿Cuándo target encoding > one-hot?

Cuando la cardinalidad es alta (>~15 valores únicos) y existe relación monotónica entre categoría y target. One-hot con 1000 zipcodes te da 1000 columnas sparse y árboles que no convergen; target encoding te da 1 sola columna numérica informativa. Siempre con CV out-of-fold para no filtrar el target.

¿Necesito escalar si uso Random Forest o XGBoost?

No. Los árboles particionan por umbrales, son invariantes a transformaciones monotónicas. Sí lo necesitás para SVM, KNN, redes, regresión lineal/logística con regularización.

¿Stratified split en regresión?

Sí — pero estratificás por una versión bucketizada del target o de una feature clave (como income_cat en California Housing). StratifiedShuffleSplit no acepta target continuo directamente.

¿StandardScaler o MinMaxScaler?

StandardScaler por default (z-score, asume distribución aprox. normal). MinMaxScaler cuando necesitás bounds fijos [0,1] (redes con sigmoide, ciertos algoritmos de visión). RobustScaler si hay outliers fuertes.

¿Hago feature engineering antes o después del split?

Features que dependen solo de la fila (ratios, log, sin/cos): antes o después, da igual. Features que dependen de estadísticos agregados (target encoding, frequency encoding, z-scores globales): siempre después del split y fit solo en train.

Referencias

- Géron, cap. 2 — End-to-End Machine Learning Project (California Housing).
- sklearn — Pipeline y ColumnTransformer
- sklearn — impute (SimpleImputer, KNNImputer, IterativeImputer)
- sklearn — TargetEncoder
- category_encoders docs
- Rubin, D. B. (1976). Inference and missing data. Biometrika.

Siguiente clase

Clase 055 — Feature Engineering avanzado: target encoding + MICE imputation

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
# Imports y configuración inicial
```

Archivos complementarios

- notebook.ipynb