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

Parte: 1 — Machine Learning Clásico · Fuente: Micci-Barreca (2001) target encoding + Van Buuren (2018) Flexible Imputation of Missing Data.  Duración estimada: 85 min.

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

> *Parte: 1 — Machine Learning Clásico · Fuente: Micci-Barreca (2001) target encoding + Van Buuren (2018) Flexible Imputation of Missing Data. Duración estimada: 85 min.*

## Objetivo

Dominar feature engineering moderno más allá de one-hot y SimpleImputer: target encoding con regularización + cross-validation (evita leakage), KNNImputer y IterativeImputer (MICE) para imputación multivariada inteligente, y category_encoders library para codificaciones modernas (CatBoost, James-Stein, hashing).

## Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Aplicar target encoding con CV (no leak): category_encoders.TargetEncoder(cv=5, smoothing=10.0).
- Aplicar KNNImputer: imputa basado en vecinos.
- Aplicar IterativeImputer (MICE) de sklearn — predice cada feature con modelo de las demás.
- Decidir entre métodos: SimpleImputer (baseline), KNN (correlaciones locales), MICE (multivariada).
- Reconocer leakage en target encoding y evitarlo con CV interno.

## Temas

- Target encoding clásico + smoothing bayesiano.
- Leak en target encoding sin CV: features ven sus propios targets.
- KNNImputer (sklearn): nearest neighbors por filas.
- IterativeImputer / MICE: estima cada feature con regresión.
- category_encoders: CatBoost, James-Stein, target, hashing.
- Pipeline-safe imputation.

## Definiciones y características

- Target encoding: reemplazar categoría con su mean target. Con smoothing: combina con global mean.
- Smoothing: enc = (n·mean_cat + k·global) / (n + k) — protege categorías raras.
- CV target encoding: cada fold usa encoding fitted en otros folds → sin leakage.
- KNNImputer: para cada NaN, promedia k vecinos en feature space.
- MICE: iterativo — predice cada feature con regresión usando las demás.
- CatBoostEncoder: variant de target encoding sin necesitar CV (ordering trick).

## Dataset / recursos

- fetch_openml('credit-g') o California Housing con NaN inyectados.
- Librerías: category_encoders (pip install category_encoders), scikit-learn, pandas.

## Ejercicios

1. Target encoding leak: encoding sobre train+test → métrica inflada. Mostrar.
2. Target encoding con CV: TargetEncoder(cv=5, smoothing=10) dentro de pipeline. Sin leak.
3. CatBoost encoder: alternativa sin CV. Comparar performance.
4. KNNImputer: con dataset con NaN, imputar con k=5; comparar contra SimpleImputer (mean).
5. MICE: IterativeImputer(estimator=BayesianRidge(), max_iter=10). Comparar.

## Homework verificable

Sobre credit-g con NaN sintéticos (10 % de missing en 3 columnas):

1. Pipeline 1: SimpleImputer + OneHot + LogReg.
2. Pipeline 2: KNNImputer + TargetEncoder(CV) + LogReg.
3. Pipeline 3: MICE + CatBoostEncoder + LogReg.
4. Comparar accuracy + tiempo.

Criterio de aceptación: pipelines modernos (2, 3) superan SimpleImputer baseline en AUC por ≥ 0.5 pp; sin target leakage en target encoding.

## Errores comunes

| Síntoma / mensaje | Causa y cómo arreglar |
|---|---|
| Target encoding fitted sobre X completo → | Leak. Fix: TargetEncoder dentro de pipelin |
| MICE con max_iter=2 poco convergente | Fix: 10-20 iterations. |
| KNNImputer lento con N grande | O(N²) en distancia. Fix: KNN con n_neighbo |
| Smoothing=0 sobre categoría con n=1 | Encoding = ese 1 target → overfit. Fix: sm |
| OneHot para 10k+ categorías | Curse of dimensionality. Fix: target encod |

## Preguntas frecuentes

Target encoding mejor que one-hot?

Para alta cardinalidad (> 50 categorías), sí — mucho. Para baja, comparable o peor.

MICE o KNN?

MICE mejor con features correlated (predice con modelo). KNN mejor con clusters locales. Probar.

Smoothing value?

Empírico — 5-30 típicamente. Más smoothing = más conservador hacia global mean.

category_encoders integrado en sklearn?

No, lib externa. sklearn tiene TargetEncoder desde 1.3, pero category_encoders tiene más options.

Para tree-based (XGBoost)?

Mucho menos crítico — los árboles manejan categóricas razonable. Pero target encoding ayuda con cardinalidad alta.

## Referencias


- Micci-Barreca (2001), A Preprocessing Scheme for High-Cardinality Categorical Attributes.
- Van Buuren (2018), Flexible Imputation of Missing Data (libro gratuito).
- category_encoders docs.
- sklearn IterativeImputer.


## Siguiente clase

Clase 056 — Selección y entrenamiento de modelo

## Apéndice: notebook (primer bloque)

Dataset sintético de clasificación con 3 categóricas de alta cardinalidad (1000 niveles) + missing values. Comparamos one-hot vs target encoding, y SimpleImputer vs MICE.

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score
from sklearn.preprocessing import OneHotEncoder
from sklearn.experimental import enable_iterative_imputer  # noqa
from sklearn.impute import IterativeImputer, SimpleImputer
from sklearn.linear_model import BayesianRidge

rng = np.random.default_rng(42)
np.random.seed(42)
```

## Archivos complementarios

- notebook.ipynb