Clase 056 — Selección y entrenamiento de modelo

Parte: 1 — Machine Learning Clásico · Fuente: Géron, cap. 2 § Select and Train a Model. · Duración estimada: 60 min.

Clase 056 — Selección y entrenamiento de modelo

> *Parte: 1 — Machine Learning Clásico · Fuente: Géron, cap. 2 § Select and Train a Model. · Duración estimada: 60 min.*

Objetivo

Que el alumno entrene varios modelos baseline sobre el pipeline ya preparado (California Housing), los compare con cross-validation en vez de un único split, identifique sub/overfitting con learning curves, y elija el candidato más prometedor para pasar a fine-tuning — sin malgastar tiempo afinando un modelo que no tiene techo.

Resultados de aprendizaje

Al finalizar la clase, el alumno podrá:

1. Entrenar baselines (LinearRegression, DecisionTreeRegressor, RandomForestRegressor) sobre el X_prepared del pipeline de la clase anterior.
2. Evaluar con cross_val_score usando K-Fold y scoring='neg_root_mean_squared_error' en vez de un solo train/test.
3. Leer learning curves para diagnosticar bias vs varianza (underfitting vs overfitting).
4. Comparar modelos con media ± desvío de los folds y decidir cuál merece HPO.
5. Persistir el modelo elegido con joblib.dump(...) para retomarlo en la próxima clase.

Temas

#	Tema	Por qué importa
1	Entrenar baseline LinearRegression y medir	Punto de comparación honesto. Si el lineal
2	DecisionTreeRegressor con RMSE = 0 en tra	Caso canónico de overfitting; te enseña a
3	cross_val_score(..., cv=10, scoring='neg_r	Estimación robusta de error de generalizac
4	RandomForestRegressor y comparación con	Baseline fuerte en tabular; suele ganar an
5	learning_curve — train vs validation score	Diagnóstico visual de bias/varianza.
6	validation_curve — score vs un hiperparáme	Antesala del grid search de la clase sigui
7	Decidir cuándo pasar a HPO vs cuándo segu	Criterio de corte; evita el agujero del tu

Definiciones y características

Baseline model

: Modelo simple (lineal, árbol corto, dummy) contra el cual se compara cualquier modelo más complejo. Si un Random Forest no le gana al LinearRegression por margen claro, el feature engineering está fallando — no el modelo.

cross_val_score(estimator, X, y, cv, scoring)

: Entrena cv veces sobre folds disjuntos y devuelve un array de scores. En sklearn, scoring siempre es "más alto es mejor" — por eso para RMSE se usa neg_root_mean_squared_error (negado) y después se invierte

el signo.

scoring

: String que selecciona la métrica. Para regresión: 'neg_root_mean_squared_error', 'neg_mean_absolute_error', 'r2'. Para clasificación: 'accuracy', 'f1', 'roc_auc'. La lista completa está en Scoring parameter.

Learning curve

: Gráfico de score (train y validation) en función del tamaño del training set. Diagnostica: brecha grande train ↔ val = overfitting (varianza alta); ambas curvas bajas y juntas = underfitting (bias alto); ambas convergen alto = modelo OK.

Validation curve

: Gráfico de score (train y validation) en función de un hiperparámetro (max_depth, n_estimators, etc.). Te muestra a ojo el rango interesante antes de tirar un grid search.

Model card

: Documento corto (markdown, una página) que registra: dataset, features, modelo, métricas en CV, hiperparámetros, fecha, supuestos y limitaciones. Práctica de Google/Hugging Face. Te salva cuando volvés al proyecto 3 meses después.

joblib.dump / joblib.load

: Serialización optimizada para objetos sklearn (matrices NumPy grandes). Preferido sobre pickle puro. Convención: modelo.pkl o modelo.joblib.

Dataset / recursos

Seguimos con California Housing y el X_prepared que sale del pipeline construido en las clases 049-050. Si arrancás esta clase fresca, el notebook regenera el pipeline desde fetch_california_housing() para que sea autocontenido.

Ejercicios

1. Baseline lineal. Entrená LinearRegression() sobre X_prepared, predecí sobre los primeros 5 ejemplos, compará con los y reales. Calculá RMSE sobre todo el train. Esperá algo en el orden de ~68k USD.

2. Árbol que memoriza. Entrená DecisionTreeRegressor(random_state=42) sin restringir profundidad. Calculá RMSE sobre train. Vas a obtener 0 (o casi). Discutí en una celda markdown por qué eso no significa que el modelo sea bueno.

3. Cross-validation honesto. Corré cross_val_score(tree, X_prepared, y, scoring='neg_root_mean_squared_error', cv=10). Reportá media y desvío del RMSE (recordá negar el signo). Compará con el lineal evaluado con el mismo cv=10.

4. Random Forest. Entrená RandomForestRegressor(n_estimators=100, random_state=42) y evaluá con CV de 10 folds. Esperá que la media baje a ~50k USD. Hacé una tabla markdown con los 3 modelos: media ± desvío.

5. Learning curve. Usá sklearn.model_selection.learning_curve sobre el Random Forest con train_sizes=np.linspace(0.1, 1.0, 5). Plotteá train_score y val_score vs tamaño. Diagnosticá: ¿alta varianza, alto bias, o convergencia?

Homework verificable

Notebook que: (a) entrene los 3 baselines sobre X_prepared; (b) corra CV de 10 folds para cada uno y guarde los arrays de scores; (c) genere una tabla comparativa con RMSE media, RMSE desvío, tiempo de fit; (d) elija el modelo más prometedor con justificación de 3 líneas en markdown; (e) plottee la learning curve del elegido; (f) serialice el modelo entrenado en models/modelo_baseline.pkl con joblib.dump.

Criterio de aceptación: El Random Forest aparece con RMSE-CV menor al lineal y al árbol pelado. El archivo .pkl se puede recargar con joblib.load y predice sin errores sobre los primeros 5 ejemplos de X_prepared.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
Reporto el RMSE como un número enorme posi	Olvidaste que sklearn devuelve scores nega
Decision tree con RMSE = 0 en train y lo d	Estás midiendo en el mismo set en que entr
cross_val_score tarda eternidad con Random	Cada fold reentrena el bosque entero. Fix:
Learning curve plana en train y val ambas	Underfitting. Fix: no es problema de datos
Hago cross_val_score sobre el X crudo (sin	Te salteaste el preprocessing. Fix: pasale

Preguntas frecuentes

¿Por qué CV de 10 y no un train/test split?

Un solo split te da un solo número con varianza alta — podés tener mala/buena suerte con esa partición. CV te da 10 números: media + desvío. Si el desvío entre folds es grande, el modelo es inestable y un score puntual miente.

¿neg_root_mean_squared_error o neg_mean_squared_error?

neg_root_mean_squared_error está disponible desde sklearn 0.22 y te devuelve directo el RMSE negado. Si tu versión es vieja, usá neg_mean_squared_error y aplicá np.sqrt(-scores) a mano.

¿Cuándo dejo de probar baselines y paso a HPO?

Cuando el mejor baseline ya le saca margen claro al segundo, y la learning curve muestra que más datos no van a mover la aguja. Si seguís en bias alto, antes de HPO sumá features o subí la familia de modelo.

¿Sirve mirar el RMSE de train?

Sí, pero solo como diagnóstico, no como métrica de selección. Train bajo + val alto = overfitting. Train alto + val alto = underfitting. La métrica que ranquea modelos es siempre la de validation (o CV).

¿Cuál es la diferencia entre learning_curve y validation_curve?

learning_curve varía el tamaño del training set (¿necesito más datos?). validation_curve varía un hiperparámetro con N fijo (¿qué max_depth conviene?). Las dos comparten el formato train-vs-val pero responden preguntas distintas.

Referencias

- Géron, cap. 2 § Select and Train a Model + § Better Evaluation Using Cross-Validation.

- sklearn cross_val_score
- sklearn learning_curve
- sklearn Scoring parameter
- Model Cards for Model Reporting (Mitchell et al., 2019)

Siguiente clase

Clase 057 — Fine-tuning: grid search y randomized search

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
# Imports y configuración inicial
```

Archivos complementarios

- notebook.ipynb