Clase 082 — Regresión con árboles

Parte: 1 — Machine Learning Clásico · Fuente: Géron, cap. 6. Duración estimada: 45 min.

Clase 082 — Regresión con árboles

Parte: 1 — Machine Learning Clásico · Fuente: Géron, cap. 6. Duración estimada: 45 min.

Objetivo

Entrenar árboles de decisión para problemas de regresión con DecisionTreeRegressor, entender por qué sus predicciones son escalonadas (constantes por hoja) y reconocer su incapacidad para extrapolar fuera del rango de entrenamiento.

Resultados de aprendizaje

Al finalizar la clase, el estudiante podrá:

- Ajustar un DecisionTreeRegressor de scikit-learn y predecir sobre datos nuevos.
- Explicar cómo el criterio MSE (squared error) decide los splits en regresión.
- Visualizar la predicción escalonada del árbol contra el target real en 1D.
- Identificar el problema de no-extrapolación y contrastarlo con regresión lineal.
- Regular max_depth / min_samples_leaf para balancear bias y varianza.

Temas

- DecisionTreeRegressor: API e hiperparámetros principales.
- Criterio de split: squared_error (MSE) y friedman_mse.
- Predicción como media del target dentro de cada hoja → función escalonada.
- Sobreajuste con árboles profundos y regularización vía max_depth, min_samples_leaf, min_samples_split.
- Limitación clave: el árbol no extrapola — predice el último valor visto en los extremos.
- Comparación rápida con regresión lineal en datos con tendencia.

Definiciones y características

- DecisionTreeRegressor: variante de árbol que en cada hoja predice un escalar (la media del target de las muestras que cayeron ahí), no una clase.
- squared_error (MSE): criterio por defecto; en cada split elige el corte que minimiza la suma ponderada de varianzas en los hijos. Es el análogo de Gini/entropía pero para regresión.
- friedman_mse: variante de MSE que ajusta una mejora propuesta por Friedman; suele dar splits levemente distintos, útil con boosting.
- Predicción escalonada (step prediction): como cada hoja devuelve una constante, la función $\hat{y}(x)$ es piecewise constant. En 1D se ve como una escalera, nunca como una curva suave.
- No extrapolación: fuera del rango [min(x_train), max(x_train)] el árbol devuelve la constante de la hoja del extremo. No hay pendiente, no hay tendencia: lineal con slope = 0 afuera.
- Regularización: max_depth limita profundidad, min_samples_leaf exige un mínimo de muestras por hoja (suaviza); sin estos límites el árbol llega a MSE = 0 en train (una hoja por muestra) y sobreajusta.
- Sensibilidad a rotaciones: igual que en clasificación, los splits son axis-aligned; si la relación es diagonal, el árbol necesita muchos cortes para aproximarla.

Dataset / recursos

- Dataset sintético 1D: x = np.linspace(-3, 3, 200), y = np.sin(x) + ruido_gaussiano(0, 0.1) — sirve para ver la escalera y la no-extrapolación.
- Alternativa real: sklearn.datasets.fetch_california_housing (regresión, 8 features) para evaluar con cross_val_score.
- Géron, Hands-On ML, cap. 6, sección "Regression" (incluye figura 6-5 con la predicción escalonada).

Ejercicios

1. Ajuste básico: generá los datos sin(x) + ruido, entrená DecisionTreeRegressor(max_depth=2) y max_depth=5, y graficá ambas predicciones sobre los puntos. Observá las escaleras.
2. MSE en train vs test: hacé train_test_split(test_size=0.3), calculá mean_squared_error en train y test para max_depth ∈ {1, 2, 4, 8, None}. ¿Dónde empieza el sobreajuste?
3. No extrapolación: predecí con el modelo entrenado sobre x_new = np.linspace(-5, 5, 200) (rango más ancho que el train). Graficá: vas a ver mesetas planas en los extremos.
4. Árbol vs lineal: entrená LinearRegression sobre los mismos datos. Comparalo con el árbol fuera del rango de entrenamiento. ¿Cuál extrapola "bien" y por qué?
5. California Housing: corré cross_val_score(DecisionTreeRegressor(max_depth=6), X, y, scoring='neg_mean_squared_error', cv=5) y compará contra max_depth=None.

Homework verificable

Entregar un script tarea_073.py que:

1. Genere y = 0.5 * x + sin(x) + ruido con x en [0, 10].
2. Entrene DecisionTreeRegressor(max_depth=4, random_state=42) y LinearRegression.
3. Prediga sobre x_new en [0, 15] (excede el rango de train).
4. Imprima el MSE en test (dentro del rango) y el valor predicho por el árbol en x=15.

Criterio de aceptación: el script corre sin error, el MSE del árbol en test es menor al de la lineal dentro del rango, y la predicción del árbol en x=15 es idéntica a su predicción en x=10 (demuestra no-extrapolación).

Errores comunes

- Esperar una curva suave: el árbol nunca produce una recta ni una curva — siempre escalones. Si necesitás suavidad, usá regresión lineal, polinómica o un ensamble como Gradient Boosting.
- Olvidar regularizar: dejar max_depth=None con pocos datos da MSE = 0 en train y MSE altísimo en test. Siempre validar con CV.
- Usar el árbol para forecasting con tendencia: si la serie tiene drift, el árbol queda clavado en el último valor del rango de train. Para tendencia, primero diferenciar o usar otro modelo.
- Confundir el criterio: criterion='gini' es para clasificación; en regresión usá squared_error o absolute_error (este último es más robusto a outliers pero más lento).
- Comparar MSE entre datasets: el MSE no es adimensional. Para comparar entre problemas usá R^2 o RMSE relativo al desvío del target.

Preguntas frecuentes

- ¿Árbol o regresión lineal? Si la relación es monótona y aproximadamente lineal y querés extrapolar,

lineal. Si hay no-linealidades, interacciones y umbrales y trabajás dentro del rango de entrenamiento, árbol (o mejor: un ensamble).

- ¿Por qué la predicción es constante por tramos? Porque cada hoja almacena un único número (la media del target del subconjunto que cayó ahí). Toda muestra que termine en esa hoja recibe el mismo valor.
- ¿Puedo usar absolute_error en lugar de MSE? Sí, optimiza la mediana por hoja en vez de la media; es más robusto a outliers pero entrena más lento y suele dar splits distintos.
- ¿Cómo elijo max_depth? Con validación cruzada sobre una grilla pequeña ([2, 4, 6, 8, 10, None]). En general entre 4 y 10 alcanza para la mayoría de datasets tabulares.
- ¿Sirve un solo árbol en producción? Casi nunca. Su varianza es alta. En la práctica se usa como bloque base de Random Forest o Gradient Boosting (próximas clases).

Referencias

- Géron, A. Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow, 3ra ed., cap. 6 — sección "Regression".
- scikit-learn: DecisionTreeRegressor.
- scikit-learn user guide: Decision Trees — Regression.

Siguiente clase

Clase 083 — Voting classifiers: hard y soft

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
# Imports y configuración inicial
```

Archivos complementarios

- notebook.ipynb