Clase 085 — Random Forests y Extra Trees

Parte: 1 — Machine Learning Clásico · Fuente: Géron, cap. 7. Duración estimada: 60 min.

Clase 085 — Random Forests y Extra Trees

Parte: 1 — Machine Learning Clásico · Fuente: Géron, cap. 7. Duración estimada: 60 min.

Objetivo

Entender Random Forests y Extra Trees como ensambles de árboles decorrelacionados, diferenciar sus mecanismos de aleatoriedad y elegir hiperparámetros razonables para problemas reales de clasificación y regresión.

Resultados de aprendizaje

Al terminar la clase vas a poder:

1. Explicar por qué un Random Forest reduce varianza respecto a un árbol único, usando los conceptos de bagging y subsampling de features.
2. Diferenciar Random Forest vs Extra Trees (thresholds óptimos vs aleatorios) y argumentar cuándo conviene cada uno.
3. Configurar los hiperparámetros clave (n_estimators, max_features, max_depth, bootstrap, min_samples_leaf) con criterios fundados.
4. Entrenar y comparar RandomForestClassifier y ExtraTreesClassifier de scikit-learn en un dataset tabular, midiendo accuracy y tiempo.
5. Interpretar oob_score_ como estimación honesta del error de generalización sin necesidad de validación cruzada.

Temas

- Repaso de bagging: bootstrap + agregación → reducción de varianza.
- Random Forest: bagging de árboles + subsampling de features en cada split.
- Extra Trees (Extremely Randomized Trees): thresholds aleatorios en lugar de óptimos.
- Hiperparámetros: n_estimators, max_features, max_depth, min_samples_leaf, bootstrap, oob_score.
- Out-of-Bag (OOB) score y su uso como reemplazo de CV.
- Comparativa empírica: bias, varianza, tiempo de entrenamiento.

Definiciones y características

- Random Forest: ensamble de árboles de decisión entrenados sobre muestras bootstrap del dataset; en cada split solo se considera un subconjunto aleatorio de features. La predicción final es voto mayoritario (clasificación) o promedio (regresión).
- Extra Trees: variante de Random Forest donde, para cada feature candidata en un split, el threshold se elige al azar en lugar de buscar el óptimo. Esto añade más aleatoriedad → mayor reducción de varianza a costa de un poco más de sesgo, y es más rápido de entrenar.
- n_estimators: cantidad de árboles. Más árboles → mejor performance y mayor estabilidad, pero rendimiento decreciente. Valores típicos: 100–500.
- max_features: cantidad de features consideradas por split. Default sqrt(n_features) para clasificación, n_features (o 1.0) para regresión. Bajarlo aumenta decorrelación entre árboles.

- bootstrap: si True (default en RF), cada árbol se entrena sobre una muestra bootstrap; si False (default en ExtraTrees), sobre todo el dataset. Solo con bootstrap=True se puede calcular OOB.
- Feature subsampling: técnica clave que diferencia RF de un simple bagging de árboles; obliga a cada árbol a explorar diferentes combinaciones, decorrelacionando sus errores.
- Decorrelación: cuanto menos correlacionados estén los errores de los árboles, mayor la reducción de varianza del ensamble (el promedio de variables correlacionadas reduce varianza más lento que el de independientes).
- OOB score (oob_score_): precisión calculada sobre las muestras que no entraron en el bootstrap de cada árbol; estima generalización sin gastar un split de validación.

Dataset / recursos

- Dataset principal: sklearn.datasets.load_breast_cancer() (569 muestras, 30 features, binario).
- Dataset secundario (regresión): sklearn.datasets.fetch_california_housing().
- Librerías: scikit-learn >= 1.3, numpy, pandas, matplotlib.
- Notebook: notebook.ipynb con ejemplos guiados.

Ejercicios

1. Baseline: entrená un DecisionTreeClassifier único sobre load_breast_cancer con random_state=42 y reportá accuracy en test (split 80/20).
2. Random Forest: entrená un RandomForestClassifier(n_estimators=200, random_state=42) y compará accuracy y tiempo contra el árbol único.
3. Extra Trees: entrená un ExtraTreesClassifier(n_estimators=200, random_state=42) y compará contra RF en accuracy y tiempo de entrenamiento.
4. OOB: entrená un RF con oob_score=True, bootstrap=True y compará oob_score_ contra el accuracy de test; ¿se parecen?
5. Sensibilidad a max_features: variá max_features en [1, 'sqrt', 'log2', 0.5, 1.0] y graficá accuracy de CV; identificá el óptimo.

Homework verificable

Sobre fetch_california_housing:

1. Entrená RandomForestRegressor y ExtraTreesRegressor con n_estimators=300, random_state=42.
2. Reportá R^2 en test (split 80/20) y tiempo de entrenamiento de cada uno.
3. Tuneá max_features con GridSearchCV (cv=3) sobre [0.3, 0.5, 0.7, 1.0] para el mejor de los dos.

Criterio de aprobación: el mejor modelo tuneado alcanza $R^2 \geq 0.80$ en test, y entregás una tabla comparativa con accuracy, tiempo y max_features óptimo.

Errores comunes

- Confundir bootstrap=False con desactivar el ensamble: los árboles siguen siendo distintos por el subsampling de features y, en ExtraTrees, por los thresholds aleatorios.
- Pedir oob_score=True con bootstrap=False: scikit-learn lanza error; el OOB requiere muestras fuera del bootstrap.
- Inflar n_estimators sin medir: pasar de 500 a 2000 árboles rara vez mueve la aguja y multiplica tiempo/memoria. Medí la curva.

- No fijar random_state: hace que los resultados no sean reproducibles entre corridas; siempre fijalo en ejercicios y benchmarks.
- Usar max_features=1.0 en clasificación: anula el subsampling de features y degrada la decorrelación; quedate con 'sqrt' salvo evidencia en contra.

Preguntas frecuentes

- ¿RF o ExtraTrees? Si tenés mucho ruido en los features o el dataset es grande y el tiempo importa, probá ExtraTrees (más rápido, más reducción de varianza). Si querés OOB score o el dataset es chico/limpio, RF suele ser un poco mejor. En la práctica: probá los dos, la diferencia suele ser < 1%.
- ¿Cuántos árboles necesito? Empezá con 100, subí a 300–500 si ves margen. Más allá, ganancias marginales.
- ¿Random Forest sufre overfitting? Mucho menos que un árbol único, pero sí puede sobreajustar con datasets muy chicos o con árboles demasiado profundos sin regularización (min_samples_leaf).
- ¿Sirven para features categóricas? Sí, pero scikit-learn requiere encoding previo (OneHot o Ordinal). Otros frameworks (LightGBM, CatBoost) los manejan nativamente.
- ¿Por qué el OOB es una buena estimación? Cada muestra queda fuera de ~37% de los árboles (por la matemática del bootstrap), lo que da un estimador casi insesgado del error de generalización, gratis.

Referencias

- Géron, A. Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow (3ra ed.), cap. 7 — Ensemble Learning and Random Forests, sección "Random Forests".
- Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 5–32.
- Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely Randomized Trees. Machine Learning, 63(1), 3–42.
- Documentación scikit-learn: RandomForestClassifier y ExtraTreesClassifier.

Siguiente clase

Clase 086 — Feature importance

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
# Imports y configuración inicial
```

Archivos complementarios

- notebook.ipynb