Clase 086 — Feature importance

Parte: 1 — Machine Learning Clásico · Fuente: Géron, cap. 7 + sklearn permutation_importance docs. Duración estimada: 70 min.

Clase 086 — Feature importance

> *Parte: 1 — Machine Learning Clásico · Fuente: Géron, cap. 7 + sklearn permutation_importance docs.*
> *Duración estimada: 70 min.*

Objetivo

Aprender a medir y comunicar qué variables aportan a un modelo basado en árboles, distinguiendo entre Mean Decrease in Impurity (MDI) —el feature_importances_ por default de scikit-learn— y permutation importance, entendiendo los sesgos de cada método. Como complemento, conocer las herramientas modernas de interpretabilidad SHAP y LIME para explicar predicciones individuales.

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Calcular feature_importances_ en un RandomForestClassifier / GradientBoostingRegressor y explicar qué mide MDI.
- Aplicar sklearn.inspection.permutation_importance sobre el set de validación y comparar el ranking con MDI.
- Identificar los sesgos de MDI (favorece features de alta cardinalidad y continuas) y de permutation (problemas con features correlacionadas).
- Generar explicaciones locales con shap.TreeExplainer e interpretar summary_plot y waterfall_plot.
- Decidir cuándo usar MDI, permutation, SHAP o LIME según contexto (auditoría, debugging, comunicación).

Temas

- Recordatorio: cómo los árboles eligen splits (impurity / variance reduction).
- MDI: suma ponderada de la reducción de impureza por feature, promediada sobre los árboles.
- Permutation importance: caída del score al permutar aleatoriamente los valores de una feature.
- Sesgos: cardinalidad alta infla MDI; correlación infla/desinfla permutation.
- MDI se calcula sobre train (riesgo de overfitting); permutation se calcula sobre test/valid.
- Interpretabilidad global vs local.
- Complemento moderno: SHAP, LIME, PDP, ICE.

Versión profundizada — 2026

El tema moderno que antes vivía como complemento dentro de esta clase ahora tiene su(s) clase(s) propia(s) con patrón completo, ejercicios y homework:

- Clase 077a — SHAP en profundidad: TreeExplainer, KernelExplainer, DeepExplainer

Definiciones y características

- feature_importances_: atributo de los estimadores tree-based de sklearn que devuelve un array

normalizado (suma 1) con la importancia MDI de cada feature.

- MDI (Mean Decrease in Impurity): para cada feature, suma de las reducciones de impureza (Gini/entropy/MSE) en cada split que la usa, ponderadas por la fracción de muestras que pasan por ese nodo, promediado sobre todos los árboles del ensemble.
- Sesgo MDI por cardinalidad: features con muchos valores únicos (IDs, continuas) ofrecen más splits candidatos y suelen aparecer infladas aunque no tengan poder predictivo real.
- Permutation importance: diferencia entre el score del modelo con la columna original y el score tras permutar aleatoriamente esa columna. Se calcula sobre datos no vistos (test/valid).
- SHAP value: contribución de una feature a la predicción de una instancia específica, promediada sobre todos los órdenes posibles de inclusión (Shapley values de teoría de juegos).
- LIME: aproximación local con un modelo interpretable (regresión lineal/árbol pequeño) entrenado sobre perturbaciones de la instancia ponderadas por cercanía.
- PDP (Partial Dependence Plot): efecto marginal promedio de una o dos features sobre la predicción, marginalizando sobre el resto.
- ICE (Individual Conditional Expectation): como PDP pero una curva por instancia, revela heterogeneidad oculta por el promedio.

Dataset / recursos

- Dataset: sklearn.datasets.fetch_california_housing (regresión) y/o load_breast_cancer (clasificación).
- Librerías: scikit-learn, shap, lime, matplotlib.
- Instalación: pip install shap lime.

Ejercicios

1. Entrená un RandomForestRegressor sobre California Housing. Imprimí feature_importances_ ordenado y graficalo como barh.
2. Calculá permutation_importance sobre el set de test con n_repeats=10. Comparalo con MDI en un DataFrame lado a lado. ¿Coincide el top-3?
3. Agregá una columna random_id = np.arange(len(X)) y reentrená. Mostrá cómo MDI le asigna importancia espuria mientras permutation la ignora.
4. Generá shap.summary_plot con TreeExplainer y explicá la diferencia entre el plot tipo beeswarm (impacto + dirección) y un bar plot de MDI.
5. Elegí una instancia mal clasificada (o con error grande en regresión) y explicala con shap.waterfall_plot. Anotá las 3 features que más empujaron la predicción.

Homework verificable

Sobre el dataset load_breast_cancer, entrená un RandomForestClassifier y entregá un notebook que:

1. Reporte el top-5 de features según MDI y según permutation (sobre test).
2. Genere shap.summary_plot y guarde el PNG.
3. Explique con SHAP una predicción correcta y una incorrecta (waterfall_plot).

Criterio de aceptación: los rankings MDI y permutation pueden diferir, pero el alumno debe justificar la diferencia en 2-3 líneas mencionando cardinalidad/correlación. El SHAP debe mostrar que las contribuciones suman (aproximadamente) la diferencia entre expected_value y la predicción.

Errores comunes

1. Interpretar MDI con features de alta cardinalidad: IDs, fechas como int, o continuas con muchos decimales aparecen infladas. Siempre validá con permutation.
2. Calcular permutation importance sobre train: si el modelo overfittea, el ranking es inútil. Siempre sobre test/valid.
3. Confiar en permutation con features correlacionadas: si dos features llevan info redundante, permutar una sola subestima ambas (el modelo se apoya en la otra). Considerá agrupar features correlacionadas.
4. Usar KernelExplainer de SHAP sobre un Random Forest: lento e innecesario. Usá TreeExplainer, que es exacto y ~100x más rápido.
5. Reportar feature importance sin escalado/contexto: un valor de 0.15 no significa nada si no decís contra qué se compara. Mostrá ranking o porcentajes acumulados.

Preguntas frecuentes

1. ¿SHAP o LIME? SHAP si trabajás con tabular y querés rigor (especialmente con árboles, por TreeExplainer). LIME si necesitás explicar texto/imagen o un modelo donde SHAP es prohibitivamente lento.
2. ¿Por qué MDI suma 1 y permutation no? MDI está normalizado por construcción; permutation devuelve la caída absoluta de score, que depende de la métrica usada.
3. ¿Puedo usar feature importance para selección de features? Sí, pero con cuidado: usá permutation sobre validación, no MDI sobre train. Mejor aún: SelectFromModel o RFE con CV.
4. ¿Qué pasa si dos features están perfectamente correlacionadas? MDI las reparte arbitrariamente; permutation puede mostrar ambas como poco importantes. SHAP también distribuye entre ellas. Detectalas con corr() antes de entrenar.
5. ¿SHAP funciona con XGBoost/LightGBM/CatBoost? Sí, los tres tienen integración nativa con TreeExplainer. De hecho, XGBoost y LightGBM exponen pred_contribs que son SHAP values calculados internamente.

Referencias

- Géron, A. Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow (3rd ed.), cap. 7 "Ensemble Learning and Random Forests" — sección "Feature Importance".
- scikit-learn docs: Permutation feature importance y partial_dependence.
- SHAP docs: <https://shap.readthedocs.io/> — Lundberg & Lee (2017), A Unified Approach to Interpreting Model Predictions, NeurIPS.
- LIME paper: Ribeiro, Singh & Guestrin (2016), "Why Should I Trust You?": Explaining the Predictions of Any Classifier, KDD. Repo: <https://github.com/marcotcr/lime>.
- Molnar, C. Interpretable Machine Learning (libro online gratuito): <https://christophm.github.io/interpretable-ml-book/>.

Siguiente clase

Clase 087 — SHAP en profundidad: TreeExplainer, KernelExplainer, DeepExplainer

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
# Imports y configuración inicial
```

Archivos complementarios

- notebook.ipynb