Lecture 12: Model Interpretability, Fairness, and Deployment

Luu Minh Sao Khue

Explainability methods (feature importance, SHAP, LIME), fairness metrics and bias mitigation, responsible/ethical ML, and the role of pipelines and deployment in trustworthy AI.

1. Learning Objectives

After this lecture, you will be able to:

- Distinguish global vs. local model interpretability.
- Compute and interpret feature importance (impurity, permutation).
- Explain Shapley values and apply SHAP for local/global insights.
- Explain LIME and fit local surrogate explanations.
- Define and compute common fairness metrics and trade-offs.
- Recognize dataset bias and discuss ethical ML principles.
- Build leakage-safe pipelines and outline deployment/monitoring.

2. Intuition

Predictive performance alone is insufficient in many domains (health, finance, justice). We need: *(i) Interpretability* (what features matter and why?), *(ii) Fairness* (are outcomes equitable across groups?), *(iii) Responsible deployment* (robustness, monitoring, redress).

Interpretability tools operate at different granularities:

- **Global:** what the model learned overall (importance, partial dependence).
- **Local:** why a particular prediction was made (SHAP, LIME, counterfactuals).

Fairness asks whether error rates or positive rates are balanced across sensitive groups (e.g. sex, race), acknowledging fundamental incompatibilities between some criteria.

Pipelines reduce *leakage* by fitting preprocessing on training only, and deployment adds versioning, monitoring, and explainability *in production*.

3. Global Interpretability: Feature Importance

3.1 Impurity-based Importance (Tree Models)

For an ensemble of decision trees, define the importance of feature j as the total impurity reduction (e.g. Gini decrease) attributed to splits on j:

$$\mathrm{Imp}(j) = \sum_{t \in \mathcal{T}} \sum_{\substack{\text{nodes } v \\ \text{split on } j}} \Delta I_t(v) \cdot \frac{n_t(v)}{n},$$

where $\Delta I_t(v)$ is impurity decrease at node v of tree t, $n_t(v)$ is samples at v, and n is total samples.

Caveats. Can be biased toward high-cardinality or continuous features; not model-agnostic.

3.2 Permutation Importance (Model-Agnostic)

Measure performance drop when a feature is randomly permuted:

$$\mathrm{PI}(j) = \mathcal{M}(f; X, y) - \mathcal{M}(f; X^{\pi(j)}, y),$$

where $\mathcal{M}$ is a metric (e.g. accuracy), and $X^{\pi(j)}$ permutes column j breaking association with y. Repeat permutations to reduce variance. Works for any fitted model.

3.3 Partial Dependence and ICE

For feature subset S,

$$\mathrm{PD}_S(x_S) = \mathbb{E}_{x_{\bar{S}}}[f(x_S, x_{\bar{S}})],$$

averaging predictions over the empirical distribution of other features. ICE (Individual Conditional Expectation) shows per-sample curves; PD is their average. *Warning:* can be misleading with strong feature interactions or correlated inputs.

4. Local Interpretability: SHAP

4.1 Shapley Values

Given a feature set $N = \{1, \ldots, p\}$ and prediction function f, the Shapley value for feature j at instance x is

$$\phi_j(f, x) = \sum_{S \subseteq N \setminus \{j\}} \frac{|S|!(p - |S| - 1)!}{p!} \Big(v(S \cup \{j\}) - v(S) \Big),$$

where $v(S)$ is the value of coalition S, often defined as the expected prediction when only features in S are known:

$$v(S) = \mathbb{E}[f(x_S, X_{\bar{S}})].$$

Core axioms: *Efficiency* ($\sum_j \phi_j = f(x) - \mathbb{E}[f(X)]$), *Symmetry*, *Dummy*, *Additivity*. These imply a unique, fair attribution under the coalition game.

4.2 SHAP Approximations

Exact computation is exponential in p. Practical variants:

- **TreeSHAP** (fast, exact for tree ensembles).
- **KernelSHAP** (model-agnostic weighted linear regression that approximates Shapley values).
- **DeepSHAP** (connections to DeepLIFT for NNs).

4.3 Global from Local

Aggregating $|\phi_j|$ across many samples yields global importance. SHAP summaries (beeswarm) show magnitude and direction of effects; dependence plots visualize interactions.

5. Local Interpretability: LIME

LIME fits a simple surrogate around a point x by sampling neighbors z and solving a locality-weighted regression/classifier:

$$\hat{g} = \arg\min_{g \in \mathcal{G}} \sum_{z \in \mathcal{Z}} \pi_x(z)\, \ell\big(f(z), g(z)\big) + \Omega(g),$$

where $\pi_x(z)$ is a proximity kernel (e.g. exponential in distance), ℓ a loss (e.g. squared loss or logistic), and Ω penalizes complexity to keep g interpretable (e.g. sparse linear model). Output coefficients explain f at x.

Caveats. Instability across runs, sensitivity to kernel and sampling; explanations are valid only locally.

6. Fairness: Definitions and Trade-offs

Let A be a sensitive attribute (e.g. group a, b), Y ground truth, $\hat{Y}$ prediction, and S score.

6.1 Group Fairness Metrics

- **Demographic Parity (DP):**
$$P(\hat{Y} = 1 \mid A = a) = P(\hat{Y} = 1 \mid A = b).$$
- **Equal Opportunity (EOpp):**
$$P(\hat{Y} = 1 \mid Y = 1, A = a) = P(\hat{Y} = 1 \mid Y = 1, A = b) \quad \text{(TPR parity)}.$$
- **Equalized Odds (EOdds):** Equalize both TPR and FPR across groups:
$$\begin{cases} P(\hat{Y} = 1 \mid Y = 1, A = a) = P(\hat{Y} = 1 \mid Y = 1, A = b), \\ P(\hat{Y} = 1 \mid Y = 0, A = a) = P(\hat{Y} = 1 \mid Y = 0, A = b). \end{cases}$$
- **Calibration within Groups:** For score S, $P(Y = 1 \mid S = s, A = a) = s$ for all s and groups.

6.2 Disparate Impact (DI)

The ratio of positive rates:

$$\mathrm{DI} = \frac{P(\hat{Y} = 1 \mid A = a)}{P(\hat{Y} = 1 \mid A = b)}.$$

A common heuristic is the "80% rule": $\mathrm{DI} \geq 0.8$.

6.3 Impossibility Results

In general, with differing base rates across groups, you cannot achieve both *calibration* and *equalized odds* simultaneously (except in degenerate cases). Trade-offs must be made.

6.4 Bias Sources and Mitigation

- **Sources:** historical bias, representation bias, measurement/label bias, selection bias, proxies for A.
- **Mitigation:** pre-processing (reweighing, sampling, repair), in-processing (fairness constraints, adversarial training), post-processing (thresholding per-group, equalized odds post-hoc).

7. Ethics and Responsible AI

7.1 Core Principles

Transparency, accountability, privacy, beneficence/non-maleficence, justice, and human oversight. Consider the right to explanation where applicable and avenues for recourse.

7.2 Dataset Considerations

Consent and licensing, de-identification, data minimization, audit trails, documentation (datasheets, model cards), and careful handling of sensitive attributes.

7.3 Risk and Harm

Assess direct harms (denied services), allocative harms, representational harms (stereotyping), and feedback loops. Plan redress mechanisms.

8. Pipelines and Deployment

8.1 Leakage-safe Pipelines

Use fit/transform pipelines to ensure preprocessing (scaling, encoding, imputation) is learned only on training folds. In K-fold CV, all preprocessing must be inside the fold loop or the library's Pipeline.

8.2 Model Packaging and Serving

Version data, code, and models; containerize (e.g. Docker), define contracts (input schemas), and implement health checks. Choose serving patterns: batch, online, streaming.

8.3 Monitoring and Drift

Track data drift (e.g. PSI), concept drift (label relationship changes), performance by segment, and *explanation drift* (e.g. SHAP distributions shifting). Set alert thresholds and retraining triggers.

8.4 Governance Artifacts

Datasheets for datasets, model cards, decision logs, approval workflow, and rollback plans.

9. Python Example: Pipeline + Importance + SHAP + LIME + Fairness

```
# End-to-end demo: classification with interpretability and fairness.
# - Pipeline: StandardScaler + RandomForest
# - Global feature importance: permutation importance
# - Local/global SHAP (TreeExplainer)
# - Local LIME explanation
# - Simple group fairness metrics (synthetic sensitive attribute)

from __future__ import annotations
import numpy as np
import pandas as pd

from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, roc_auc_score
from sklearn.inspection import permutation_importance

# SHAP and LIME (install if missing: pip install shap lime)
import shap
from lime.lime_tabular import LimeTabularExplainer

# 1) Data ---------------------------------------------------------------
data = load_breast_cancer()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = pd.Series(data.target, name="target")

# Create a synthetic sensitive attribute A.
A = (X["mean texture"] > X["mean texture"].median()).astype(int)
A.name = "group"  # 0 vs 1

X_train, X_test, y_train, y_test, A_train, A_test = train_test_split(
    X, y, A, test_size=0.25, random_state=42, stratify=y
)

# 2) Pipeline and training ----------------------------------------------
```

```
pipe = Pipeline([
    ("scaler", StandardScaler(with_mean=True, with_std=True)),
    ("rf", RandomForestClassifier(
        n_estimators=300, max_depth=None, random_state=42
    )),
])
pipe.fit(X_train, y_train)

# Basic performance
proba_test = pipe.predict_proba(X_test)[:, 1]
y_pred = (proba_test >= 0.5).astype(int)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("ROC-AUC:", roc_auc_score(y_test, proba_test))

# 3) Global: permutation importance ---------------------------------------
pi = permutation_importance(
    pipe, X_test, y_test, n_repeats=10, random_state=42, scoring="roc_auc"
)
imp_df = pd.DataFrame({
    "feature": X.columns,
    "mean_importance": pi.importances_mean,
    "std": pi.importances_std
}).sort_values("mean_importance", ascending=False)
print("\nTop permutation importances:\n", imp_df.head(10))

# 4) Local+Global: SHAP (TreeSHAP) ----------------------------------------
scaler = pipe.named_steps["scaler"]
rf = pipe.named_steps["rf"]

X_train_tr = scaler.transform(X_train)
X_test_tr = scaler.transform(X_test)

explainer = shap.TreeExplainer(rf)
idx_bg = np.random.default_rng(0).choice(len(X_train_tr), 200, replace=False)
shap_values = explainer.shap_values(X_test_tr, check_additivity=False)

mean_abs_phi = np.mean(np.abs(shap_values[1]), axis=0)
shap_imp = pd.DataFrame({
    "feature": X.columns,
    "mean_abs_shap": mean_abs_phi
}).sort_values("mean_abs_shap", ascending=False)
print("\nTop SHAP importances:\n", shap_imp.head(10))

# 5) Local: LIME explanation for a single instance ----------------------
explainer_lime = LimeTabularExplainer(
    training_data=X_train.values,
    feature_names=list(X.columns),
    class_names=list(data.target_names),
    discretize_continuous=True,
    verbose=False,
    mode="classification"
)
```

```
i = 0  # explain the first test instance
def predict_fn(arr: np.ndarray) -> np.ndarray:
    return pipe.predict_proba(pd.DataFrame(arr, columns=X.columns))

lime_exp = explainer_lime.explain_instance(
    data_row=X_test.iloc[i].values,
    predict_fn=predict_fn,
    num_features=8
)
print("\nLIME local explanation:")
for feat, w in lime_exp.as_list():
    print(f"{feat}: {w:.3f}")

# 6) Fairness metrics (group-based) ------------------------------------
def tpr(y_true: np.ndarray, y_hat: np.ndarray) -> float:
    tp = np.sum((y_true == 1) & (y_hat == 1))
    fn = np.sum((y_true == 1) & (y_hat == 0))
    return tp / (tp + fn + 1e-12)

def fpr(y_true: np.ndarray, y_hat: np.ndarray) -> float:
    fp = np.sum((y_true == 0) & (y_hat == 1))
    tn = np.sum((y_true == 0) & (y_hat == 0))
    return fp / (fp + tn + 1e-12)

def positive_rate(y_hat: np.ndarray) -> float:
    return np.mean(y_hat == 1)

mask_a0 = (A_test == 0).values
mask_a1 = (A_test == 1).values

tpr0, tpr1 = tpr(y_test[mask_a0], y_pred[mask_a0]), tpr(y_test[mask_a1], y_pred[mask_a1])
fpr0, fpr1 = fpr(y_test[mask_a0], y_pred[mask_a0]), fpr(y_test[mask_a1], y_pred[mask_a1])
pr0, pr1   = positive_rate(y_pred[mask_a0]), positive_rate(y_pred[mask_a1])

print("\nFairness metrics (synthetic groups):")
print(f"TPR group0: {tpr0:.3f}, group1: {tpr1:.3f}, TPR: {abs(tpr0-tpr1):.3f}")
print(f"FPR group0: {fpr0:.3f}, group1: {fpr1:.3f}, FPR: {abs(fpr0-fpr1):.3f}")
print(f"PosRate g0: {pr0:.3f}, g1: {pr1:.3f}, DI (g1/g0): {(pr1/(pr0+1e-12)):.3f}")
```

10. Practical Tips

- **Use model-agnostic explainability:** Prefer permutation importance, KernelSHAP, or LIME when comparing different model families. These methods can be applied regardless of model structure.
- **Combine global and local explanations:** Global methods (e.g., feature importance) reveal overall influence of features, while local methods (SHAP/LIME) explain individual predictions. Combine both for comprehensive understanding.
- **Stability and robustness:** Validate explanation stability by repeating SHAP or

LIME with different seeds, background samples, and perturbations. Report average importance and confidence intervals.

- **Fairness monitoring:** Always evaluate model performance separately across sensitive groups (e.g., gender, age, ethnicity). Track group-wise metrics such as TPR, FPR, and Disparate Impact to detect bias.
- **Avoid data leakage:** Always include preprocessing (scaling, encoding, imputation) inside a `Pipeline` or `ColumnTransformer`. This ensures that transformations are fitted only on training folds.
- **Ethical transparency:** Document datasets and models using *datasheets* and *model cards*. State clearly the intended use, limitations, and possible ethical risks.
- **Deployment readiness:** When deploying models, version-control the code, data, and model artifacts. Log model predictions, confidence scores, and SHAP explanations for traceability and auditing.
- **Monitor model drift:** Continuously monitor performance and explanation drift (change in top SHAP features or feature importances). Establish retraining triggers when drift exceeds defined thresholds.
- **Human-in-the-loop:** Combine automated explanations with expert feedback. Domain experts should review model decisions, especially in high-stakes applications.
- **Visual communication:** Use interpretable visualizations such as SHAP summary plots, feature dependence plots, and fairness dashboards to communicate results to non-technical stakeholders.

11. Summary

We explored interpretability and fairness — essential dimensions of trustworthy machine learning. Global interpretability explains how a model behaves overall (e.g., feature importance, partial dependence), while local interpretability explains individual predictions through methods like SHAP and LIME. Fairness ensures that models perform equitably across demographic groups and do not amplify social biases.

We also discussed ethical considerations and the importance of transparency, accountability, and privacy in AI systems. Finally, we introduced the concepts of pipelines, deployment, and model monitoring as practical tools for maintaining model reliability and fairness in real-world settings.

- **Interpretability:** Understand what features drive predictions globally and locally using SHAP, LIME, and feature importance.
- **Fairness:** Evaluate and balance group fairness metrics such as Demographic Parity, Equal Opportunity, and Equalized Odds.
- **Ethics:** Integrate transparency, privacy, and accountability into model development and deployment.
- **Pipelines and Deployment:** Use end-to-end pipelines to avoid data leakage; containerize models and track performance and drift after deployment.

12. Exercises

1. **Shapley Axioms:** Derive and prove the efficiency axiom for Shapley values, showing that the sum of all feature contributions equals the model output difference $f(x) - \mathbb{E}[f(X)]$.

2. **Permutation vs. Impurity Importance:** Construct a synthetic dataset where a categorical feature with many levels appears more important under impurity-based importance than permutation importance. Explain why this happens.

3. **Local Explanation Stability:** Run LIME multiple times on the same instance with different random seeds. Quantify the variance in feature weights and discuss its implications for reliability.

4. **Fairness Trade-off:** Show mathematically why calibration and equalized odds cannot be satisfied simultaneously when groups have different base rates.

5. **Pipeline Validation:** Compare cross-validation accuracy with and without using `Pipeline`. Demonstrate how fitting a scaler on full data causes data leakage and overestimates performance.

6. **Monitoring Drift:** Define thresholds for data drift (e.g., Population Stability Index > 0.2) and explanation drift (change in top SHAP features). Simulate a drift scenario and design a retraining alert strategy.

7. **Ethics Reflection:** Pick one real-world case (e.g., facial recognition, medical diagnosis) where lack of explainability caused public concern. Discuss what interpretability or fairness tools could have mitigated the issue.