Pedagogical Report

Teaching Fairness, Explainability, and Threshold-Based Mitigation in Machine Learning

Srujan Voodarla

1. Teaching Philosophy

My teaching philosophy is built around three core pillars: learning by doing, computational skepticism, and transparent, interpretable modeling practices. In the domain of fairness and ethical machine learning, students do not learn effectively by reading definitions alone. Instead, they must *interact directly* with biased models, interpret their behavior, and use technical tools to mitigate harm.

This module immerses students in a full lifecycle:

1. Construct synthetic data with controllable bias
2. Train a modern ML classifier
3. Interpret decisions using SHAP
4. Quantify disparities with fairness metrics
5. Mitigate bias using multiple techniques
6. Explore thresholds interactively through a Streamlit app

This reflects an experiential pedagogy: students build intuition by observing how small changes in data or thresholds produce large downstream effects.

Constructivist Learning

Students learn by constructing mental models through:

- Coding exercises
- Visualizations
- Interactive exploration (Streamlit sliders)
- Interpretation of real plots and metrics

They continuously revise hypotheses each time they interact with the system.

Computational Skepticism

In line with course themes, the project trains students to **question model performance**, ask whether predictions are equitable, and investigate whether mitigation genuinely reduces harm or simply hides the issue.

Ethical and Practical Relevance

Real systems—credit scoring, hiring tools, healthcare triage—face fairness constraints. This project simulates such environments, giving students **industry-relevant experience** with:

- SHAP explainability
- Bias detection & mitigation
- Performance–fairness trade-offs
- Interactive decision-boundary tuning

This aligns with my goal of preparing students to be responsible practitioners capable of deploying ML systems with awareness and accountability.

2. Target Audience & Background Assumptions

Intended Learners

- Graduate students in data science, ML, computer science, analytics
- Practitioners entering fields involving responsible AI or risk assessment
- Students familiar with the basics of supervised learning

Assumed Background

Students should already know:

- Python and pandas
- Standard ML workflows (train/test split, classification models)
- Basic metrics (accuracy, confusion matrix, ROC curve)

Not Required

Students do *not* need:

- Prior experience with fairness literature
- Deep mathematical knowledge of Shapley values
- Experience building dashboards or apps

The instructional design introduces each concept in manageable steps.

3. Learning Objectives

By the end of this module, students will be able to:

Knowledge Outcomes

- Explain how fairness metrics quantify bias
- Understand SHAP's mathematical foundation and interpretability value
- Describe threshold-based and reweighing mitigation

Technical Outcomes

- Generate synthetic hiring datasets with controlled demographic bias
- Train an XGBoost classifier and evaluate its performance
- Compute SHAP explanations (global + local)
- Compare feature influence across demographic groups
- Build and interact with a Streamlit fairness dashboard

streamlit_app

Critical Reasoning Outcomes

- Analyze the trade-offs between fairness and model accuracy
- Evaluate the limitations of different mitigation techniques
- Diagnose bias propagation from data → model → predictions

Communication Outcomes

- Present explanation results clearly
- Use visualizations (Plotly, SHAP) to justify claims
- Interpret and communicate threshold effects interactively

4. Pedagogical Approach & Rationale

The module uses the **Explain → Show → Try** model, which aligns with active learning principles.

Explain

We begin with conceptual foundations:

- What is algorithmic bias?
- How do Shapley values distribute contributions?
- Why do thresholds affect positive prediction rates?
- What do fairness metrics measure?

Show

Students observe:

- SHAP summary and force plots
- Disparate impact and demographic parity calculations
- ROC curves, confusion matrices
- Probability distributions by gender in the Streamlit dashboard
- Real-time threshold adjustments and fairness outcomes

The Streamlit app visualizes phenomena that are otherwise abstract.

Try

Students perform:

- Hands-on exercises
- Scenario A vs Scenario B comparisons

- Threshold tuning to minimize DPD
- Debugging tasks (TreeExplainer → KernelExplainer fallback)
- Sensitivity experiments with dataset size and seed

This gradual release strengthens retention and confidence.

5. Concept Deep Dive

5.1 Shapley Values and SHAP

SHAP assigns each feature a contribution score $\phi_i$to a prediction:

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N|-|S|-1)!}{|N|!} [f(S \cup \{i\}) - f(S)]$$

Where:

- N= set of all features
- $f(\cdot)$= model prediction

Key properties:

- **Efficiency:** Contributions sum to the prediction difference
- **Symmetry:** Equal features receive equal contributions
- **Monotonicity:** If feature impact grows, SHAP increases

Why SHAP?

- Works for nonlinear models (XGBoost)
- Provides both global and local explanations
- Connects mathematically to cooperative game theory

Limitations

- Exact computation exponential; approximations needed
- KernelExplainer can be slow
- Correlated features complicate attribution

5.2 Fairness Metrics

Demographic Parity Difference (DPD)

$$DPD = P(\hat{Y} = 1 \mid A = priv) - P(\hat{Y} = 1 \mid A = unpriv)$$

Disparate Impact Ratio (DIR)

$$DIR = \frac{P(\hat{Y} = 1 \mid unpriv)}{P(\hat{Y} = 1 \mid priv)}$$

Values < 0.8 indicate potential discrimination.

Equal Opportunity Difference (EOD)

$$EOD = TPR_{priv} - TPR_{unpriv}$$

These metrics highlight different aspects of fairness:

- DPD → selection rate equality
- DIR → proportionality
- EOD → equal true positive performance

5.3 Threshold-Based Mitigation

Fairness often depends not just on model probabilities, but on how those probabilities are **converted to decisions**.

The Streamlit app demonstrates:

- Male and female thresholds can be adjusted independently
- Probability distributions differ between groups
- Positive-rate curves intersect at fairness-optimal points
- Confusion matrices evolve with thresholds
- ROC curve remains unchanged (threshold-invariant)

This illustrates *post-processing* fairness mitigation with no model retraining.

5.4 Connection to Course Themes

GIGO (Garbage In, Garbage Out)

Synthetic penalties applied at data generation propagate through:

- training,
- SHAP explanations,
- predictions.

Botspeak Framework

Students must articulate:

- what the model is doing,
- why certain features matter,
- how thresholding alleviates disparity.

Computational Skepticism

The entire module teaches that:

- High accuracy ≠ fairness
- Models inherit bias from data
- Mitigation must be measured, not assumed

5.5 Real-World Workflow Alignment

The module mirrors real ML systems in:

- Hiring automation
- Risk scoring
- Credit decisioning
- Healthcare triage

The **Streamlit dashboard** simulates how practitioners explore thresholds and fairness impact before operational deployment.

6. Implementation Analysis

6.1 Architecture Overview

Jupyter Notebook Workflow

1. Generate synthetic data
2. Train model
3. Compute SHAP explanations
4. Evaluate fairness
5. Apply mitigation
6. Compare scenarios

Streamlit App Workflow

streamlit_app

1. Generate dataset dynamically
2. Train XGBoost
3. Compute predicted probabilities
4. Let users adjust male/female thresholds
5. Recompute:
    - DPD
    - positive rates
    - confusion matrix
6. Visualize:
    - probability distributions
    - positive-rate curves
    - thresholds (vertical lines)

    - ROC curve

6.2 Libraries & Tools

- **XGBoost:** high-performance tabular modeling
- **SHAP:** interpretability framework
- **Plotly:** interactive visualization
- **Streamlit:** real-time exploration and teaching tool
- **scikit-learn:** utilities for splitting, metrics
- **pandas/numpy:** data manipulation

These tools mirror real industry stacks.

6.3 Performance Considerations

- TreeExplainer can fail depending on version; KernelExplainer fallback needed
- Streamlit remains performant due to small dataset sizes (1000–10000)
- Plotly renders efficiently in browser
- Threshold sweeps (200 points) are vectorized

6.4 Edge Cases & Limitations

- SHAP values unstable under multicollinearity
- Different fairness metrics can conflict
- Thresholding may improve fairness at accuracy cost
- Synthetic datasets don't fully capture real hiring biases
- ROC is threshold-invariant, may mislead fairness intuitions

7. Assessment & Effectiveness

7.1 Learning Assessment

Evaluation is based on:

- Exercises (metric computation, SHAP interpretation)
- Debugging tasks (TreeExplainer error resolution)
- Notebook replication
- Streamlit threshold tuning demonstrations
- Written explanations in the pedagogical report

7.2 Common Student Difficulties

- Misinterpreting SHAP plots
- Believing accuracy implies fairness
- Confusing DPD vs DIR
- Understanding why thresholds affect fairness but not ROC

7.3 Support for Different Learning Styles

- **Visual learners:** SHAP summary plots, Plotly histograms, positive-rate curves
- **Hands-on learners:** threshold sliders, code exercises
- **Reading/writing learners:** tutorial + report
- **Auditory learners:** video walkthrough explanation

8. Future Improvements & Extensions

- Add adversarial debiasing neural nets
- Include causal fairness analysis (counterfactual fairness)
- Add intersectional group fairness metrics
- Enable exporting threshold settings from Streamlit
- Add full logging and experiment tracking
- Support multi-class demographic categories

Self-Assessment

I believe my project falls in the top 25% of submissions. I justify this placement based on the following points:

Technical Depth

I implemented a complete bias-analysis workflow including dataset generation, SHAP explainability, fairness metrics, two bias scenarios, and two mitigation methods. I also built an interactive Streamlit app, which goes beyond standard requirements.

Pedagogical Quality

My materials include a structured tutorial, a detailed pedagogical report, a 10-minute teaching script, exercises, solutions, and a debugging guide. The content clearly follows the Explain → Show → Try model and supports multiple learning styles.

Professional Documentation

The project repository is fully organized with a clean folder structure, README, requirements, modular source code, and a reproducible notebook.

Alignment With Course Themes

The project strongly integrates GIGO principles, computational skepticism, and explainability practices, directly reflecting the goals of INFO 7390.

Overall, due to its completeness, clarity, interactivity, and real-world relevance, I assess my submission as meeting the standard of the highest-performing projects in the class.

LINKS

Link to the YouTube video:

https://youtu.be/mxklTPfNQhU

Link to the GitHub Repo:

SRUJAN730/TAKE-HOME-FINAL