EAI Validator v1.3.0
Comprehensive API and CLI Documentation

WHIS

August 11, 2025

Contents

1 Overview

Ethical AI (EAI) Validator is a Python package for auditing machine learning systems for bias, fairness, and regulatory compliance. Version 1.3.0 introduces:

[leftmargin=1.2em]Structured inclusion of training scenario and hyperparameters in PDF reports Hyperparameter Impact Analysis with risk and rationale Optional Feature Contribution Disparities (SHAP-backed when available) Configurable bias/fairness thresholds via validator config and/or per-report criteria CLI flags: `--scenario`, `--hyperparameters` (JSON), `--output`, `--verbose`

1.1 Installation

```
pip install whis-eai
```

1.2 Basic Usage

```
from eai_validator import EAValidator
validator = EAValidator(config={'bias_threshold': 0.1, 'fairness_threshold': 0.8})
predictions = [1,0,1,0,1,0,1,0]
true_labels = [1,0,1,0,1,0,1,0]
protected_attributes = {
  'gender': ['male','female','male','female','male','female','male','female'],
  'age_group': ['young','old','young','old','young','old','young','old'],
}
bias_report = validator.audit_bias(predictions, true_labels, protected_attributes)
fairness_metrics = validator.calculate_fairness_metrics(predictions,
    protected_attributes)
metadata = {
  'model_name': 'MyModel',
  'scenario': 'Experiment-1',
  'hyperparameters': {'max_depth': 10},
  'bias_report': bias_report,
  'fairness_metrics': fairness_metrics
}
audit_criteria = {'bias_threshold': 0.12, 'fairness_threshold': 0.85}
report_path = validator.generate_compliance_report(metadata, audit_criteria,
    output_path='example_reports/my_report.pdf')
```

2 Command-Line Interface

2.1 Synopsis

```
whis-eai [--version] [--config PATH] [--output PATH] [--verbose]
               [--scenario TEXT] [--hyperparameters JSON]
```

2.2 Options

Flag	Description
`--version`	Print CLI version (v1.3.0) and exit.

`--config PATH`	Reserved for future configuration file support.
`--output PATH`	If provided, generates a minimal PDF report embedding scenario and hyperparameters.
`--verbose`	Print initialization and additional details to stdout.
`--scenario TEXT`	Scenario name/description displayed in CLI and included in PDF when `--output` is used.
`--hyperparameters JSON`	JSON dictionary of hyperparameters to display and embed.

2.3 Examples

```
# Display scenario/hyperparameters only
whis-eai --scenario "Exp-1" --hyperparameters '{"max_depth": 12, "C": 1.0}' --verbose
# Generate report directly
whis-eai --scenario "Exp-1" \
  --hyperparameters '{"max_depth": 12, "n_estimators": 200}' \
  --output "example_reports/cli_report.pdf"
```

3 Python API

This section documents `eai_validator.eai_validator.EAValidator` and its convenience functions.

3.1 `class EAValidator(config: dict = None)`

Main validator implementing bias detection, fairness assessment, PDF reporting, monitoring, and disparity resolution.

Constructor

Parameter	Description
`config`	Optional dict. Recognized keys: `bias_threshold` (float, default 0.1), `fairness_threshold` (float, default 0.8).

3.1.1 `audit_bias(predictions, true_labels, protected_attributes)`

Detect bias across protected attributes. Computes per-group metrics including accuracy, precision, recall, F1, positive rate, statistical parity, equalized odds, demographic parity, and a bias score. Inputs:

[leftmargin=1.2em]`predictions`: 1D array-like (binary or multiclass) `true_labels`: 1D array-like, same length as predictions `protected_attributes`: dict of `str -> list/array`, each list same length as predictions

Returns a `pandas.DataFrame` with one row per group per attribute. Raises: `ValueError` on length mismatches or empty protected attributes.

3.1.2 `calculate_fairness_metrics(predictions, protected_attributes)`

Compute fairness metrics across protected attributes. Returns:

[leftmargin=1.2em]`overall_metrics`: positive rate, total samples `protected_attribute_metrics`: per-attribute group metrics `fairness_scores`: per-attribute fairness score $\in [0, 1]$ (higher is fairer)

Returns a dict.

3.1.3 `generate_compliance_report(metadata, audit_criteria, output_path=None)`

Generate a structured PDF compliance report. Sections include:

[leftmargin=1.2em]Report Info and Model Info Training Scenario and Hyperparameters (sorted table) Bias Analysis (if `metadata['bias_report']` provided) Fairness Assessment (if `metadata['fairness_metrics']` provided) Feature Contribution Disparities (optional, if `metadata['feature_disparities']`) Hyperparameter Impact Analysis with risk and rationale Likely Contributing Factors (summarizes worst fairness/bias and top suspect hyperparameters) Overall Compliance Summary table GDPR and AI Act compliance tables Recommendations

Threshold Resolution:

[leftmargin=1.2em]Start with defaults (`bias` 0.1, `fairness` 0.8) Override from `audit_criteria` if provided Override from `self.config` if present

Parameters:

Name	Description
`metadata`	Dict including keys like `model_name`, optional `scenario`, `hyperparameters`, `bias_report` (DataFrame), `fairness_metrics` (dict), optional `feature_disparities` (dict)
`audit_criteria`	Dict with `bias_threshold` (float) and `fairness_threshold` (float)
`output_path`	Optional path to save the PDF; otherwise a timestamped name in CWD

Returns a str.

3.1.4 `compute_feature_disparities(model, X, protected_attributes, max_features=10, sample_size=1000)`

Estimate which features contribute most to disparities for each protected attribute.

[leftmargin=1.2em]If SHAP is available: uses `shap.Explainer` or `shap.TreeExplainer` on a sample of rows Fallback: importance-weighted (feature importance or coefficients) differences of absolute group means

Returns a dict `attr -> list of {feature, disparity, top_group, bottom_group}`. Can be embedded into the PDF via `metadata['feature_disparities']`.

3.1.5 `hyperparameter_ablation(model, X_train, y_train, X_test, y_test, protected_attributes, params_to_probe=None, max_variants_per_param=2)`

Empirically probes selected hyperparameters (sklearn-style estimators) by training nearby variants and comparing average fairness:

[leftmargin=1.2em]Computes `baseline_avg_fairness` from the given model For each parameter in `params_to_probe` (defaults to a common set if present), evaluates up to `max_variants_per_param` alternative values Returns detailed `impacts` and a ranked `summary` by worst delta vs baseline

Returns a dict.

3.1.6 monitor_realtime(predictions_stream)

Simulated batch monitoring; emits alerts when positive rates deviate beyond bias_threshold from 0.5. Returns a list of alert dicts; stores monitoring history. Returns a list.

3.1.7 suggest_resolutions(bias_report)

Generate resolution suggestions based on statistical parity, equalized odds, demographic parity, and bias score. Returns suggestions, priority, and effort. Returns a dict.

3.2 Convenience Functions

Available at package level in eai_validator:

[leftmargin=1.2em]audit_bias(predictions, true_labels, protected_attributes) calculate_fai protected_attributes) generate_compliance_report(metadata, audit_criteria, output_path=No monitor_realtime(predictions_stream) suggest_resolutions(bias_report)

4 eai Submodules

4.1 eai.compliance

[leftmargin=1.2em]generate_compliance_report(metadata, audit_criteria): Simple report generator (legacy, reportlab) check_gdpr_compliance(model_metadata): Returns a basic GDPR compliance dict, score, and recommendations check_ai_act_compliance(model_metadata) Same for EU AI Act requirements generate_compliance_summary(gdpr_results, ai_act_results): Combined summary with overall score and compliance level

4.2 eai.fairness

[leftmargin=1.2em]calculate_fairness_metrics(predictions, protected_attributes, label_encoders=None) fairness_score(predictions, protected_attributes, label_encoders=No Average fairness across attributes; returns 1.0 if no scores available calculate_parity_metrics(predicti true_labels, protected_attributes): Simplified parity metrics assess_fairness(predictions, protected_attributes, threshold=0.8): Overall fairness assessment with recommendations

4.3 eai.monitoring

[leftmargin=1.2em]class RealTimeMonitor(config=None): monitor(predictions_stream), reset(), get_history() monitor_realtime(predictions_stream, config=None): Convenience wrapper

5 Data Contracts and Edge Cases

[leftmargin=1.2em]Arrays/lists must be 1D and of equal length for predictions/labels/protected values. Very small groups (e.g., < 5 or < 10 samples depending on function) are skipped to avoid unstable metrics. Threshold precedence: audit_criteria $>$ self.config $>$ defaults. hyperparameter_ablation expects estimators compatible with sklearn's get_params/fit/predict.

6 Performance Notes

[leftmargin=1.2em]Monitoring and bias calculation are vectorized with NumPy; typical workloads (10k rows) complete within a few seconds on a standard laptop. SHAP explanations can be expensive; limit `sample_size` in `compute_feature_disparities`.

7 End-to-End Example (Concise)

```
validator = EAValidator(config={'bias_threshold': 0.1, 'fairness_threshold': 0.8})
bias_report = validator.audit_bias(preds, labels, protected)
fairness = validator.calculate_fairness_metrics(preds, protected)
# Optional: disparities
feature_disp = validator.compute_feature_disparities(model, X_test, {'gender':
    gender_test})
metadata = {
  'model_name': 'MyModel',
  'scenario': 'A/B test',
  'hyperparameters': model.get_params(),
  'bias_report': bias_report,
  'fairness_metrics': fairness,
  'feature_disparities': feature_disp
}
criteria = {'bias_threshold': 0.12, 'fairness_threshold': 0.85}
validator.generate_compliance_report(metadata, criteria, output_path='example_reports/
    ab.pdf')
```

8 Versioning

Version **1.3.0**. See CHANGELOG.md for detailed release notes.