Individual Fairness in AutoML

Tara Sabooni* Yasaman Samadzadeh†

Abstract

Machine learning models must balance predictive accuracy with individual fairness, which requires that similar individuals receive similar predictions regardless of sensitive attributes. We operationalize individual fairness through counterfactual consistency and frame fairness-aware model selection as a multi-objective hyperparameter optimization problem. Using Automated Machine Learning (AutoML) techniques, specifically Sequential Model-based Algorithm Configuration (SMAC), we automatically identify Pareto-optimal configurations representing different accuracy–fairness trade-offs. Our experiments on the Adult Income dataset with multiple model types (Random Forest, MLP, and SenSeI) demonstrate that SMAC successfully identifies diverse Pareto-optimal configurations, revealing that significant fairness improvements can be achieved with relatively small accuracy sacrifices. We evaluate two experimental paradigms: models trained with sensitive features and models trained without sensitive features using proxy-based evaluation. Results reveal distinct trade-off profiles: Random Forest provides balanced performance achieving near-perfect consistency (>99%) with competitive accuracy, MLP prioritizes accuracy, and SenSeI prioritizes fairness. These findings suggest that careful hyperparameter optimization can yield substantial fairness benefits without requiring specialized fairness-aware training algorithms.

1 Introduction

Machine learning systems deployed in high-stakes applications must ensure fair treatment of individuals. While traditional **group fairness** metrics measure disparities across demographic groups, they have limitations: a model can satisfy group fairness while still discriminating against individual cases. **Individual fairness** addresses this by requiring that similar individuals receive similar predictions regardless of sensitive attributes [2]. We operationalize individual fairness through **counterfactual consistency** [5], which measures whether predictions remain stable when sensitive attributes are perturbed (e.g., flipping sex from male to female).
However, achieving both high accuracy and high counterfactual consistency presents a fundamental trade-off: models optimized for accuracy often exploit correlations between sensitive attributes and outcomes, violating individual fairness [6]. This raises the question: *How can we systematically identify model configurations that optimally balance accuracy and fairness?* Traditional approaches involving manual hyperparameter tuning may not yield optimal trade-offs. We address this by framing fairness-aware model selection as a **multi-objective hyperparameter optimization problem** and leveraging Automated Machine Learning (AutoML) techniques, specifically Sequential Model-based Algorithm Configuration (SMAC) [10], to automatically identify Pareto-optimal configurations representing different accuracy–fairness trade-offs.

*tara.sabooni@studio.unibo.it
†yasaman.samadzadeh@studio.unibo.it

2 Data

We conduct our experiments on the **Adult Income** dataset, also known as the Census Income dataset [8]. This dataset is a standard benchmark in fairness-aware machine learning and provides a realistic binary classification task for evaluating the trade-off between predictive performance and individual fairness.
The dataset is derived from the 1994 U.S. Census database and contains demographic and economic information about individuals. The prediction task is to determine whether an individual's annual income exceeds $50,000. Prior to preprocessing, the dataset consists of 48,842 instances with 14 original features, including demographic attributes (e.g., age, sex, race), employment-related variables (e.g., workclass, occupation, hours per week), and economic indicators (e.g., capital gain and capital loss).
Two sensitive attributes are particularly relevant for fairness analysis: **sex** (male/female) and **race**, with categories including White, Black, Asian-Pac-Islander, Amer-Indian-Eskimo, and Other. These attributes are used to evaluate counterfactual consistency, a key notion of individual fairness.

2.1 Preprocessing

We apply a comprehensive preprocessing pipeline to prepare the dataset for model training and evaluation. The main steps are summarized below.

Target Variable The target variable is in binary format, where `>50K` is encoded as 1 (positive class) and `≤50K` as 0 (negative class). The positive class represents approximately 24.78% of the data, indicating a moderate class imbalance that motivates our use of balanced accuracy as an evaluation metric (see Section 4)

Missing Value Handling All rows containing missing values are removed. This results in the removal of approximately 6,620 instances (about 13.5% of the original dataset), leaving 45,222 samples for subsequent experiments.

Feature Type Classification Features are classified as categorical or numerical based on OpenML metadata. Notably, in the OpenML version of the Adult dataset, several originally continuous features (including `age`, `capital-gain`, `capital-loss`, and `hours-per-week`) are provided in pre-binned categorical form. Overall, the dataset contains 11 categorical features and 3 numerical features.

Feature Removal We remove the `fnlwgt` feature, which represents a census sampling weight and is not intended for predictive modeling. Additionally, the `education` feature is dropped due to redundancy with `education-num`, which encodes the same information numerically. After this step, 12 features remain.

One-Hot Encoding All categorical features are transformed using one-hot encoding, expanding each categorical variable with k categories into k binary indicator features. This includes both sensitive attributes (`sex` and `race`) and non-sensitive attributes. The `relationship` feature is also one-hot encoded, as it is later used as a proxy variable in one of our experimental settings.

Sensitive Attribute Representation Sensitive attributes are handled as follows after one-hot encoding:

- **Sex**: The attribute is represented by two binary features, `sex_Male` and `sex_Female`. To avoid multicollinearity (since `sex_Male` = 1 - `sex_Female`), only `sex_Male` is retained in the final feature set.
- **Race**: The attribute is represented by binary indicators for each racial category (`race_White`, `race_Black`, `race_Asian-Pac-Islander`, `race_Amer-Indian-Eskimo`, `race_Other`) . All race categories are retained in the experiments.

High-Cardinality Feature Removal The `native-country` feature contains 42 distinct categories and results in a highly sparse representation after one-hot encoding. To reduce dimensionality and mitigate overfitting, all features with the prefix `native-country_` are removed.

Feature Scaling Numerical features, specifically `education-num`, are standardized using z-score normalization to achieve zero mean and unit variance. One-hot encoded binary features are left unscaled, as they are already in a normalized 0/1 format.

2.2 Final Dataset

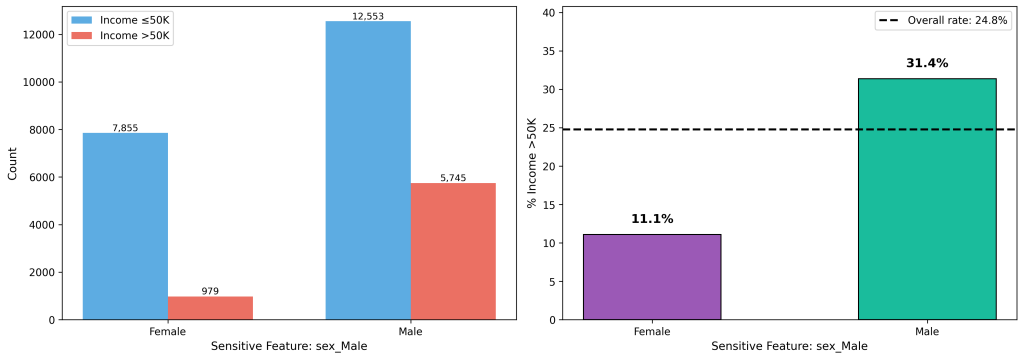
After preprocessing, the final dataset consists of 45,222 samples and 62 features, comprising binary one-hot encoded categorical variables and one standardized numerical feature. The sensitive attributes (`sex_Male` and race indicators) are retained in the model input to enable direct counterfactual fairness evaluation.

2.3 Sensitive Attribute Distribution

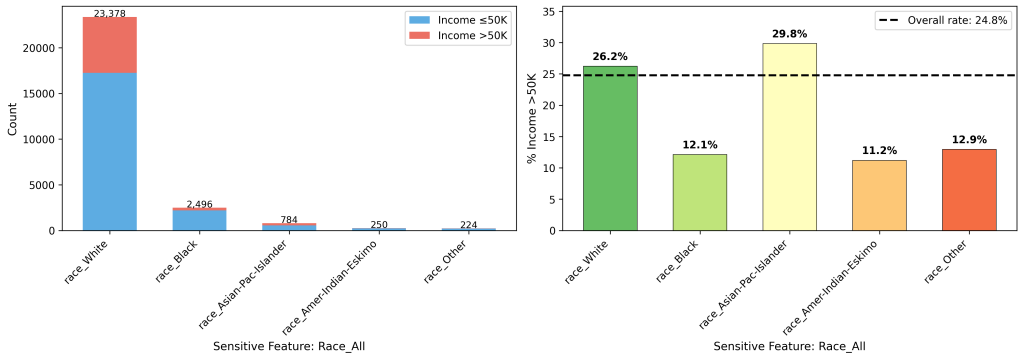
Figure 1a and Figure 1b show the distribution of samples across sensitive groups.
The dataset exhibits class imbalance: only 24.8% of samples belong to the positive class (income >$50K), motivating our use of balanced accuracy. Additionally, the base rate of high-income individuals varies across sensitive groups—for example, 31.4% for males versus 11.1% for females. While these disparities exist in the data, our fairness objective is not to equalize outcomes across groups, but rather to ensure that a model's prediction for an individual does not depend on their sensitive attributes.

3 Models

We evaluate three machine learning models within our multi-objective fairness optimization framework: **Random Forest**, **Multi-Layer Perceptron** (MLP), and **SenSeI** (Sensitive Set Invariance). These models represent three distinct learning paradigms: ensemble-based methods (Random Forest), neural networks (MLP), and fairness-aware in-training algorithms (SenSeI). For each model, we define a hyperparameter configuration space that is explored using multi-objective optimization.



(a) Distribution by sex.



(b) Distribution by race.

Figure 1: Sensitive attribute distributions in Adult Income dataset. Left: sample counts by income class. Right: percentage of high-income individuals per group.

3.1 Random Forest

Random Forest is an ensemble learning method that constructs a collection of decision trees during training and aggregates their predictions via majority voting for classification. Each tree is trained on a bootstrap sample of the data, and at each split only a random subset of features is considered. This stochasticity promotes diversity among trees, reduces variance, and mitigates overfitting while preserving strong predictive performance. The hyperparameter configuration space for Random Forest is reported in Table 1.

Table 1: Random Forest hyperparameter configuration space.

Hyperparameter	**Range / Values**	**Description**
`n_estimators`	$[10, 200]$	Number of trees in the ensemble.
`max_depth`	$[3, 20]$	Maximum depth of each decision tree.
`min_samples_split`	$[2, 20]$	Minimum samples required to split an internal node.
`min_samples_leaf`	$[1, 10]$	Minimum samples required at a leaf node.
`criterion`	{gini, entropy}	Split quality measure.
`max_features`	{sqrt, log2, None}	Number of features considered at each split.

3.2 Multi-Layer Perceptron

The Multi-Layer Perceptron (MLP) is a feedforward neural network composed of an input layer, one or more fully connected hidden layers, and an output layer producing class probabilities for binary classification. Non-linear activation functions enable the model to capture complex relationships in the data. The network is trained using backpropagation with gradient-based optimization to minimize the cross-entropy loss. The hyperparameter configuration space for MLP is reported in Table 2. Early stopping is employed with a validation fraction of 0.1 and a maximum of 500 training iterations to reduce overfitting and limit training time.

Table 2: MLP hyperparameter configuration space.

Hyperparameter	**Range / Values**	**Description**
`n_hidden_layers`	$[1, 3]$	Number of hidden layers.
`n_neurons`	$[16, 256]$ (log)	Neurons per hidden layer.
`activation`	{relu, tanh, logistic}	Hidden-layer activation function.
`solver`	{adam, sgd}	Optimization algorithm.
`alpha`	$[10^{-5}, 10^{-1}]$ (log)	L2 regularization strength.
`learning_rate_init`	$[10^{-4}, 10^{-1}]$ (log)	Initial learning rate.

3.3 SenSeI (Sensitive Set Invariance)

SenSeI [4] (Sensitive Set Invariance) is an in-training individual fairness algorithm that enforces fairness constraints during model optimization. Unlike post-processing methods, SenSeI directly integrates fairness considerations into the learning objective. Our implementation follows the PyTorch-based version provided by IBM's *inFairness* framework [3].

3.3.1 Sensitive Features Removed from Model Input

A key design principle of SenSeI is that sensitive attributes (e.g., sex or race) are *completely removed* from the prediction model's input. As a result, the classifier never directly accesses sensitive information. However, non-sensitive features may still act as proxies for sensitive attributes. For example, in the Adult Income dataset, the `relationship` feature (e.g., `Wife` or `Husband`) is strongly correlated with sex and can enable indirect discrimination.
To address this issue, SenSeI uses sensitive attributes only during training, not as predictive inputs, to learn a fairness-aware distance metric. This allows the algorithm to identify individuals who differ primarily due to sensitive attributes or their proxies and penalize unequal treatment of such individuals.
In our experiments, SenSeI is applied in *Approach 2*, where sensitive attributes are excluded from the input feature set but provided separately for fairness metric learning.

3.3.2 Fairness Mechanism and Training Procedure

SenSeI enforces individual fairness through a two-stage process:

1. **Learning a Fair Distance Metric**: SenSeI learns a distance metric that identifies directions in feature space corresponding to sensitive attribute variations. Specifically, a logistic

regression model is trained to predict sensitive attributes (e.g., sex, race) from the non-sensitive features. The learned coefficients define a *sensitive subspace*—the directions along which changes in features are most predictive of sensitive attributes. The fair distance metric then measures similarity while projecting out this sensitive subspace, treating individuals as "close" if they differ primarily along sensitive-correlated dimensions.

2. **Adversarial Fairness Regularization**: An adversarial auditor searches for pairs of individuals that are close under the learned fair distance metric but receive different predictions. When such unfair pairs are detected, the model is penalized via a transport-based regularizer. The training alternates between the auditor identifying unfairness and the model updating its parameters to reduce both prediction error and fairness violations.

The final training objective balances predictive accuracy with a fairness regularization term that discourages disparate treatment of individuals who are similar in non-sensitive aspects.

3.3.3 Counterfactual Evaluation Using Proxy Features

Since sensitive attributes are not part of the model input, counterfactual consistency for SenSeI is evaluated using proxy features. In particular, the `relationship` feature is used as a proxy for sex, where `relationship_Wife` and `relationship_Husband` correspond to female and male proxies, respectively. Counterfactuals are generated by flipping these proxy features, and prediction consistency is measured accordingly. The hyperparameter configuration space for SenSeI is reported in Table 3.
The hyperparameter configuration space for SenSeI includes:

Table 3: SenSeI hyperparameter configuration space.

Hyperparameter	Range / Values	Description
`n_hidden_layers`	$[1, 2]$	Number of hidden layers in the neural network.
`n_neurons`	$[50, 150]$ (log)	Number of neurons per hidden layer.
`learning_rate`	$[5\times10^{-4}, 5\times10^{-3}]$ (log)	Learning rate for the Adam optimizer.
`rho`	$[2.0, 15.0]$ (log)	Strength of the fairness regularization.
`eps`	$[0.05, 0.2]$ (log)	Margin parameter for the fairness constraint.
`auditor_nsteps`	$[10, 30]$	Number of adversarial auditor steps.
`auditor_lr`	$[10^{-3}, 5\times10^{-2}]$ (log)	Learning rate for the auditor.
`batch_size`	$[64, 256]$ (log)	Batch size used during training.
`epochs`	$[3, 8]$	Number of training epochs.

The configuration space is adapted from the default recommendations to ensure computational tractability while preserving the effectiveness of the fairness constraints.

4 Metrics

We evaluate our models using a multi-objective framework that jointly considers predictive performance and individual fairness. Specifically, we use **balanced accuracy** to assess classification performance and **counterfactual consistency** to quantify individual fairness. For

each hyperparameter configuration, models are trained on the training set and evaluated on a held-out validation set. A separate test set is reserved for final evaluation of the selected Pareto-optimal configurations.

4.1 Balanced Accuracy

Balanced accuracy is employed to measure predictive performance in the presence of class imbalance. Unlike standard accuracy, balanced accuracy equally weights the performance on both classes by averaging their recall values. For binary classification, balanced accuracy is defined as

$$\text{Balanced Accuracy} = \frac{\text{TPR} + \text{TNR}}{2}, \tag{1}$$

where

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{2}$$

$$\text{TNR} = \frac{\text{TN}}{\text{TN} + \text{FP}}. \tag{3}$$

Here, TP, TN, FP, and FN denote the numbers of true positives, true negatives, false positives, and false negatives, respectively.
Balanced accuracy takes values in the range $[0, 1]$, where 1 corresponds to perfect classification performance and 0.5 corresponds to random guessing in the binary setting. This metric is particularly appropriate for the Adult Income dataset, which exhibits class imbalance, with approximately 24.78% of instances belonging to the positive class.
For multi-objective optimization, we convert balanced accuracy into an error metric that is minimized:

$$E = 1 - \text{Balanced Accuracy}. \tag{4}$$

4.2 Counterfactual Consistency

Counterfactual consistency is used as our primary metric for evaluating individual fairness. It measures the extent to which a model's predictions remain invariant when sensitive attributes are modified, while all other features are held fixed. A model is considered more individually fair if its predictions are stable under such counterfactual interventions.
The computation of counterfactual consistency differs depending on whether the sensitive attribute is binary or multi-categorical.

4.2.1 Binary Sensitive Attributes

For binary sensitive attributes (e.g., `sex_Male`), counterfactual consistency is computed by flipping the binary value ($0 \leftrightarrow 1$) for each instance and comparing the resulting predictions:

$$\text{CC}_{\text{binary}} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\left[y_{\text{pred}}^{(i)} = y_{\text{flip}}^{(i)} \right], \tag{5}$$

where n denotes the number of test instances, $y_{\text{pred}}^{(i)}$ is the model prediction for instance i using the original features, $y_{\text{flip}}^{(i)}$ is the prediction obtained after flipping the sensitive attribute, and $\mathbb{1}[\cdot]$ is the indicator function.

4.2.2 Multi-Categorical Sensitive Attributes

For multi-categorical sensitive attributes represented using one-hot encoding (e.g., `race` with k categories), counterfactual consistency is computed by evaluating all possible alternative category assignments. For an instance i belonging to category $c^{(i)}$, we generate $k-1$ counterfactuals by setting each of the remaining categories to 1 while setting all others, including the original category, to 0.
The per-instance consistency is defined as

$$\text{Consistency}^{(i)} = \frac{1}{k-1} \sum_{j \neq c^{(i)}} \mathbb{1}\left[y^{(i)}_{\text{pred}} = y^{(i)}_{\text{flip},j} \right], \tag{6}$$

where $y^{(i)}_{\text{flip},j}$ denotes the prediction for instance i when the sensitive attribute is counterfactually set to category j.
The overall counterfactual consistency for the dataset is then computed as the mean over all instances:

$$\text{CC}_{\text{multi}} = \frac{1}{n} \sum_{i=1}^{n} \text{Consistency}^{(i)} = \frac{1}{n(k-1)} \sum_{i=1}^{n} \sum_{j \neq c^{(i)}} \mathbb{1}\left[y^{(i)}_{\text{pred}} = y^{(i)}_{\text{flip},j} \right]. \tag{7}$$

This formulation ensures that fairness is evaluated across all possible counterfactual changes of the sensitive attribute, providing a comprehensive assessment of individual fairness for multi-category attributes.
Counterfactual consistency takes values in the range $[0, 1]$, where 1 indicates perfect consistency (predictions never change under counterfactual interventions) and 0 indicates complete inconsistency.
For multi-objective optimization, we convert counterfactual consistency into an inconsistency metric that is minimized:

$$I = 1 - \text{Counterfactual Consistency}. \tag{8}$$

5 Methodology

We formulate fairness-aware machine learning as a **multi-objective hyperparameter optimization problem** that simultaneously optimizes predictive performance and individual fairness. Our approach leverages Automated Machine Learning (AutoML) techniques, specifically Sequential Model-based Algorithm Configuration (SMAC), to efficiently search the hyperparameter space and identify Pareto-optimal configurations that represent different trade-offs between accuracy and counterfactual consistency.

5.1 Multi-Objective Optimization Framework

We frame the problem of finding fair and accurate models as a multi-objective optimization task with two competing objectives:

1. **Minimize Error**: $E(\theta) = 1 - \text{Balanced Accuracy}(\theta)$, where θ represents a hyperparameter configuration

2. **Minimize Inconsistency**: $I(\theta) = 1 - \text{Counterfactual Consistency}(\theta)$

Since these objectives are typically in tension—improving one often worsens the other—there is no single optimal solution. Instead, we seek the **Pareto front**, the set of all configurations that are not dominated by any other configuration.

Definition 1 (Pareto Dominance)**.** Configuration θ_j **dominates** configuration θ_i, denoted $\theta_j \prec \theta_i$, if:

$$\forall k \in \{1,2\} : f_k(\theta_j) \leq f_k(\theta_i) \quad \wedge \quad \exists k \in \{1,2\} : f_k(\theta_j) < f_k(\theta_i) \tag{9}$$

where $f_1 = E$ (error) and $f_2 = I$ (inconsistency) are the two objectives to minimize.

The Pareto front provides a set of optimal trade-offs, allowing practitioners to select configurations based on their specific requirements for accuracy and fairness.

5.2 Sequential Model-based Algorithm Configuration (SMAC)

We employ **SMAC** [10], a state-of-the-art AutoML framework for hyperparameter optimization, to efficiently search the hyperparameter space. SMAC uses sequential model-based optimization (SMBO) to intelligently select which hyperparameter configurations to evaluate, making it particularly well-suited for computationally expensive objective functions like model training and fairness evaluation.

5.2.1 SMAC Algorithm Overview

SMAC operates in iterations, where each iteration consists of the following steps:

1. **Model Training**: Train a probabilistic model (random forest) on the history of evaluated configurations and their objective values.
2. **Acquisition Function Optimization**: Use the probabilistic model to identify promising configurations through an acquisition function that balances exploration (trying uncertain regions) and exploitation (focusing on promising regions).
3. **Configuration Evaluation**: Evaluate the selected configuration(s) by training the model and computing both objectives (error and inconsistency).
4. **Update History**: Add the evaluated configuration and its objectives to the run history.

This process continues until a termination criterion is met (either a time limit or a maximum number of evaluations). The algorithm maintains a probabilistic model that captures the relationship between hyperparameters and objectives, allowing it to make informed decisions about which configurations to evaluate next.

5.3 Multi-Objective Optimization with ParEGO

To handle the multi-objective optimization problem introduced in Section 5, we employ **ParEGO** [9] (Pareto Efficient Global Optimization), which transforms the multi-objective problem into a series of single-objective problems through scalarization. At each iteration, ParEGO samples different weight vectors that determine the relative importance of each objective, then uses SMAC to optimize the resulting scalarized objective. By sampling different weights across iterations, ParEGO naturally explores different regions of the Pareto front (defined in Section 5), ensuring diverse solutions are found. This approach allows SMAC's efficient search to be applied directly to multi-objective optimization.

5.4 Evaluation Pipeline

Our evaluation pipeline follows a train/validation/test split approach, where models are trained on the training set, hyperparameters are optimized using the validation set, and final evaluation is performed on the held-out test set.

5.4.1 Data Splits

We split the dataset into three sets:

- **Training Set** (60%): Used to train models during hyperparameter optimization
- **Validation Set** (20%): Used by SMAC to evaluate hyperparameter configurations
- **Test Set** (20%): Held out for final evaluation and not used during optimization

All splits use stratified sampling to preserve the class distribution across sets. The validation set ensures that hyperparameter optimization is based on a held-out evaluation, preventing overfitting to the training data.

5.4.2 Configuration Evaluation Function

For each hyperparameter configuration θ, SMAC calls an evaluation function $f(\theta) = [E(\theta), I(\theta)]$ that computes both objectives. The evaluation procedure is summarized in Algorithm 1.

Algorithm 1 Configuration Evaluation Function

Require: Hyperparameter configuration θ, training set $(X_{\text{train}}, y_{\text{train}})$, validation set $(X_{\text{val}}, y_{\text{val}})$, sensitive column index(es)
Ensure: Error $E(\theta)$, Inconsistency $I(\theta)$
1: Train model M_θ on $(X_{\text{train}}, y_{\text{train}})$
2: $\hat{y} \leftarrow M_\theta.\text{predict}(X_{\text{val}})$
3: $\text{Acc} \leftarrow \text{BalancedAccuracy}(y_{\text{val}}, \hat{y})$
4: $X_{\text{flip}} \leftarrow \text{FlipSensitiveAttributes}(X_{\text{val}})$
5: $\hat{y}_{\text{flip}} \leftarrow M_\theta.\text{predict}(X_{\text{flip}})$
6: $\text{Cons} \leftarrow \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[\hat{y}_i = \hat{y}_{\text{flip},i}]$
7: **return** $E(\theta) = 1 - \text{Acc}$, $I(\theta) = 1 - \text{Cons}$

This evaluation function is computationally expensive, requiring: (1) training a model from scratch (the dominant cost), (2) multiple prediction passes for counterfactual evaluation (especially for multiclass attributes with $k - 1$ flips), and (3) computing both objectives on the validation set. SMAC's surrogate model predicts which configurations are promising, avoiding evaluation of likely poor configurations and focusing computational resources on promising regions of the hyperparameter space.

5.4.3 Binary vs. Multiclass Counterfactual Evaluation

The evaluation function adapts based on the type of sensitive attribute (see Section 4 for the mathematical formulation). For binary attributes (e.g., sex), counterfactual consistency requires one additional prediction pass over the validation set. For multiclass attributes (e.g., race), the exhaustive approach requires $k - 1$ additional prediction passes (where k is the number of categories), making it more computationally expensive but providing a more comprehensive assessment of individual fairness.

5.5 Configuration Spaces

We define separate hyperparameter configuration spaces for each model type, enabling SMAC to search over model-specific hyperparameters. The configuration spaces are defined using ConfigSpace, which provides a flexible framework for specifying hyperparameter search spaces. Detailed descriptions of the hyperparameters for each model type (Random Forest, MLP, and SenSeI) are provided in Section 3.

The configuration spaces are extremely large, making exhaustive search computationally infeasible. This highlights the importance of efficient search methods like SMAC, which can identify good configurations with hundreds to thousands of evaluations rather than billions.

5.6 Pareto Front Extraction and Analysis

After SMAC optimization completes, we extract the Pareto-optimal configurations from the run history. A configuration is Pareto-optimal if no other evaluated configuration dominates it according to Definition 1. We identify the Pareto front by comparing all pairs of configurations to determine dominance relationships, which represents the set of optimal trade-offs between accuracy and fairness.

5.6.1 Knee-Optimal Point Selection

While the Pareto front provides multiple optimal configurations, practical applications often require selecting a single model for deployment. To identify a balanced configuration that offers a good trade-off between accuracy and fairness, we employ the **knee-optimal point selection method**.

The knee point is defined as the configuration on the Pareto front with maximum perpendicular distance from the line connecting the two extreme points: the configuration with highest accuracy (lowest error) and the configuration with highest consistency (lowest inconsistency). This geometric criterion identifies the "elbow" of the Pareto front, where the trade-off curve exhibits maximum curvature. Intuitively, the knee point represents the configuration where improving one objective requires the largest sacrifice in the other, making it a natural choice for balanced performance.

In our experiments, we use the knee-optimal configuration for detailed evaluation, including fairness-accuracy confusion matrix analysis and test set performance assessment (see Section 6). This selection method ensures that we evaluate models that balance both objectives rather than favoring accuracy or fairness exclusively.

5.7 Experimental Approaches

We conduct experiments using two distinct approaches to evaluate how sensitive attributes are handled:

5.7.1 Approach 1: Models with Sensitive Features

In Approach 1, we train standard models (Random Forest and MLP) with sensitive features included in the input. Counterfactual consistency is evaluated by directly flipping the sensitive attributes in the validation set. This approach tests whether models trained with sensitive features can learn to be fair despite having access to sensitive information.

Evaluation: For binary sensitive attributes (e.g., sex), we flip the binary value ($0 \leftrightarrow 1$). For multiclass attributes (e.g., race), we use an exhaustive approach that tests consistency across all possible category reassignments.

5.7.2 Approach 2: Models without Sensitive Features + SenSeI

In Approach 2, we train models without sensitive features in the input, evaluating whether removing sensitive attributes prevents discrimination. We also include SenSeI [3], an in-training fairness algorithm that uses sensitive features separately to learn a fair distance metric. Counterfactual consistency is evaluated using proxy features (e.g., `relationship_Wife` as a proxy for sex) since sensitive features are not available in the model input.
Motivation: This approach addresses the concern that models may still discriminate through proxy features even when sensitive attributes are removed. SenSeI explicitly enforces fairness during training using the sensitive features (provided separately) to learn a distance metric that ignores sensitive attribute variations.

5.8 Implementation and Computational Considerations

Our implementation uses SMAC v2.0+ [10] for hyperparameter optimization, ConfigSpace for defining hyperparameter spaces, scikit-learn for Random Forest and MLP models, and inFairness [3] with PyTorch for SenSeI. All experiments are conducted with fixed random seeds to ensure reproducibility.
The computational cost of our methodology is dominated by the evaluation function, which requires training a model for each hyperparameter configuration. SMAC's efficiency comes from reducing the number of evaluations needed compared to grid search or random search, typically finding good configurations with 10–100x fewer evaluations [11].

6 Results

6.1 Approach 1: Models with Sensitive Features

In this approach, we train Random Forest (RF) and Multi-Layer Perceptron (MLP) models with sensitive features included in the training data. We use SMAC for multi-objective hyperparameter optimization, simultaneously maximizing balanced accuracy and counterfactual consistency. We evaluate fairness with respect to two sensitive attributes: sex (binary) and race (multiclass with 5 categories).

6.1.1 Pareto Front Analysis

Figure 2 and Figure 3 show the Pareto fronts obtained from the multi-objective optimization for sex and race, respectively. Each point represents a hyperparameter configuration, with Pareto-optimal configurations connected by dashed lines. The knee-optimal point (marked with $\star$) represents the configuration offering the best trade-off between accuracy and fairness.
For the sex attribute, both RF and MLP achieve high consistency (>95%) while maintaining competitive balanced accuracy. The trade-off is more pronounced for race, where the exhaustive multiclass evaluation (checking consistency across all 5 categories) imposes a stricter fairness requirement.

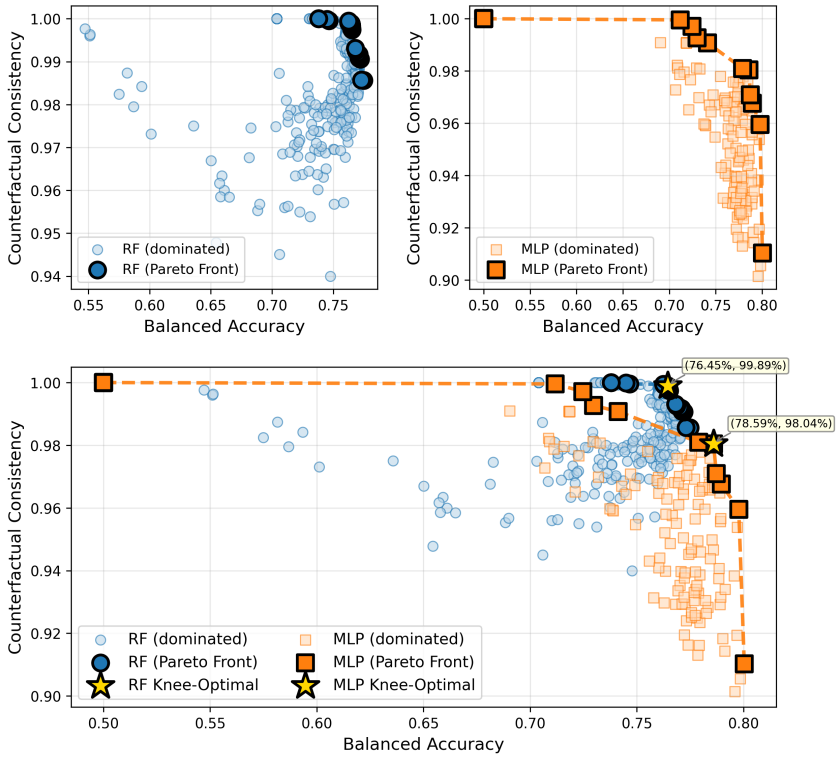


Figure 2: Pareto front for sex (binary). Top: individual model fronts. Bottom: combined comparison with knee-optimal points marked.

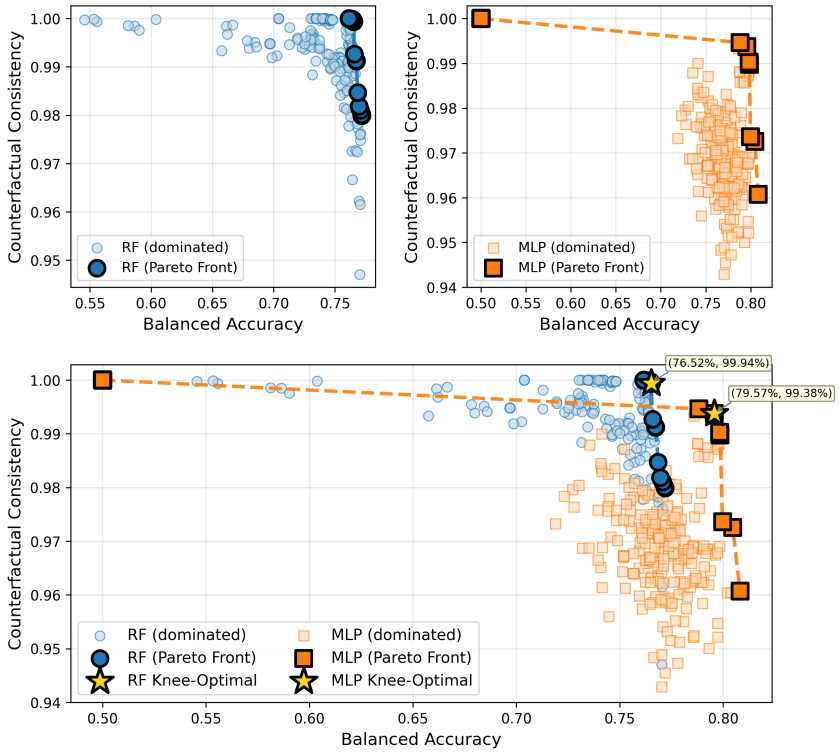
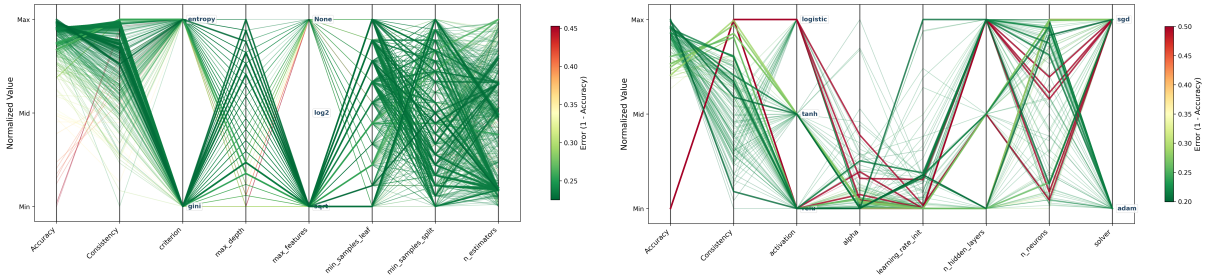


Figure 3: Pareto front for race (multiclass). The stricter consistency requirement (prediction must be unchanged across all 5 racial categories) results in lower overall consistency compared to the binary sex attribute.

A consistent pattern emerges across both sensitive attributes: Random Forest tends to achieve higher accuracy but lower counterfactual consistency, while MLP exhibits the opposite behavior—lower accuracy but higher consistency. This suggests that RF is more prone to learning decision boundaries that exploit sensitive attribute information, whereas MLP's smoother decision boundaries may be inherently less sensitive to single-feature perturbations.

6.1.2 Hyperparameter Analysis

Figures 4 shows parallel coordinate plots illustrating how hyperparameter configurations map to the optimization objectives. Bold lines indicate Pareto-optimal configurations, while faint lines represent dominated solutions. The color gradient reflects the error rate (1 - accuracy).



(a) Random Forest (b) MLP

Figure 4: Parallel coordinate plots for sex attribute showing hyperparameter-to-objective mappings.

Several patterns emerge from the Pareto-optimal configurations:

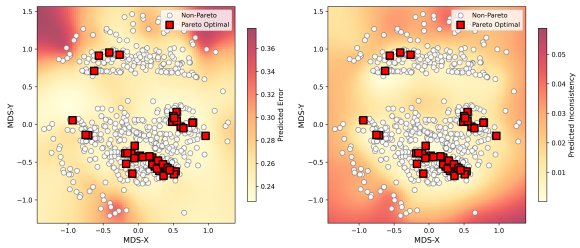
Random Forest. The Pareto front shows a preference for `criterion=gini` and `max_features=sqrt`, with notably no Pareto-optimal configurations using `log2`. Additionally, higher values of `min_samples_leaf` appear more frequently among optimal configurations, suggesting that increased regularization helps balance accuracy and fairness.

MLP. The optimal configurations concentrate on lower values of learning rate and regularization strength (alpha), indicating that slower, more controlled learning produces better trade-offs between accuracy and counterfactual consistency.

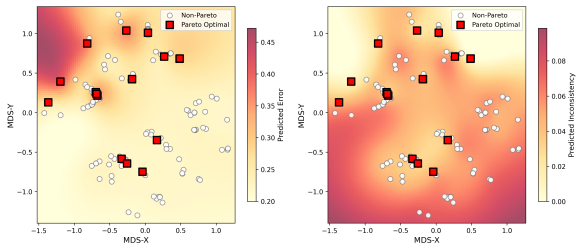
6.1.3 Configuration Space Visualization

Figures 5 and 6 present MDS (Multidimensional Scaling) projections of the hyperparameter search space. MDS reduces the high-dimensional configuration space to 2D while preserving pairwise distances between configurations—similar hyperparameter settings appear close together, while dissimilar ones are far apart.[1] Each plot shows two panels: the left displays the interpolated error landscape, and the right displays the interpolated inconsistency landscape. Red squares indicate Pareto-optimal configurations; white circles are dominated solutions.
Across all plots, Pareto-optimal configurations tend to cluster in specific regions of the hyperparameter space rather than being scattered uniformly, suggesting that optimal trade-offs are

[1]Categorical hyperparameters (e.g., activation function, criterion) are converted to integer indices (ordinal features) before computing distances. While this treats categories as ordinal, it provides a reasonable approximation for visualization purposes.

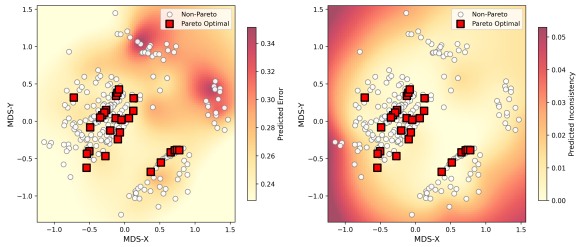


(a) Random Forest

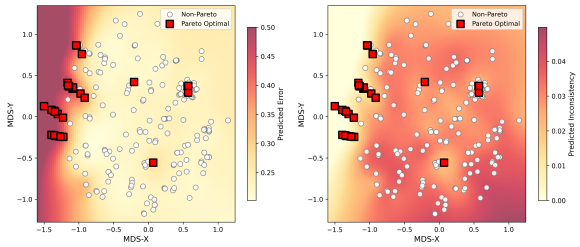


(b) MLP

Figure 5: MDS projection for sex attribute. Left panel: error landscape. Right panel: inconsistency landscape.



(a) Random Forest



(b) MLP

Figure 6: MDS projection for race attribute. Left panel: error landscape. Right panel: inconsistency landscape.

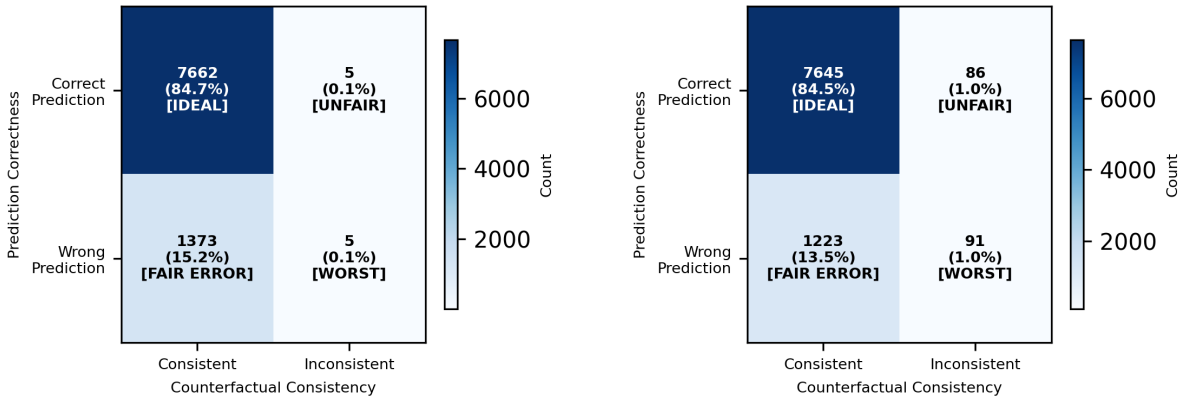
achieved by similar hyperparameter combinations. In Figures 5a, 5b, and 6a, the Pareto-optimal points predominantly fall within yellow regions in both panels, indicating configurations that achieve low error and low inconsistency simultaneously.

6.1.4 Fairness-Accuracy Confusion Matrix

To understand the relationship between prediction correctness and counterfactual consistency, we analyze the fairness-accuracy confusion matrix for the knee-optimal configurations. This matrix partitions samples into four categories:

- **Correct + Consistent [IDEAL]**: The model predicts correctly, and the prediction remains unchanged when the sensitive attribute is flipped. This is the desired outcome.
- **Correct + Inconsistent [UNFAIR]**: The model predicts correctly, but the prediction would change if the sensitive attribute were different. This represents "right for the wrong reasons"—the model relies on sensitive information.
- **Wrong + Consistent [FAIR ERROR]**: The model predicts incorrectly, but at least it treats similar individuals consistently regardless of their sensitive attribute.
- **Wrong + Inconsistent [WORST]**: The model is both incorrect and unfair—the worst possible outcome.

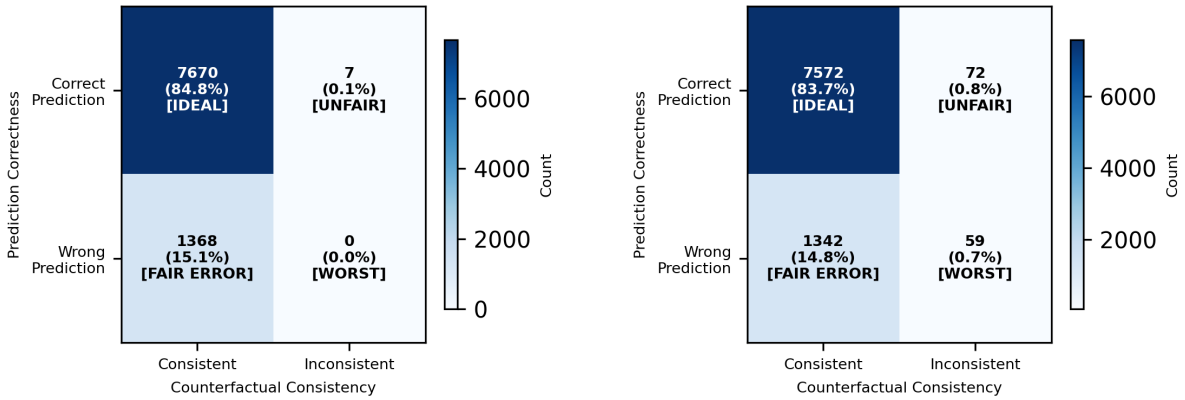
Several patterns emerge from the confusion matrices. First, the vast majority of samples fall into the ideal quadrant (Correct + Consistent), with over 84% in all cases. Second, RF consistently achieves higher consistency than MLP (99.9% vs 98.0% for sex; similar pattern for race), while MLP achieves slightly higher accuracy. This aligns with our earlier observation from the Pareto



(a) RF - Sex

(b) MLP - Sex

Figure 7: Fairness-accuracy confusion matrices for sex (knee-optimal models).



(a) RF - Race

(b) MLP - Race

Figure 8: Fairness-accuracy confusion matrices for race (knee-optimal models).

front analysis. Third, the "right but unfair" rate (Correct + Inconsistent) is very low across all models (<1.1%), indicating that the knee-optimal configurations successfully minimize reliance on sensitive attributes. Finally, correct predictions tend to be more consistent than incorrect ones, suggesting that fairness and accuracy are positively correlated in these models.

6.1.5 Test Set Evaluation

Throughout the previous analyses, models were trained on the training set and evaluated on the validation set for hyperparameter optimization and selection. Here, we present the final evaluation on the held-out test set, which was not used during any part of the optimization process.

The test set results are consistent with our validation set findings. RF achieves higher counterfactual consistency (99.87–99.91%) compared to MLP (98.01–98.56%), while MLP achieves slightly higher balanced accuracy (77.34–78.76% vs 75.92–76.05%). This confirms the trade-off pattern observed in the Pareto front analysis: RF configurations prioritize fairness, whereas MLP configurations lean toward predictive performance. Importantly, the small gap between validation and test performance indicates that the knee-optimal models generalize well and do not overfit to the validation set.

Table 4: Test set performance for Approach 1 (knee-optimal configurations)

Sensitive Attr.	Model	Accuracy (%)	Balanced Acc. (%)	Consistency (%)
Sex	RF	84.20	75.92	**99.87**
	MLP	**84.64**	**77.34**	98.01
Race	RF	**84.36**	76.05	**99.91**
	MLP	84.06	**78.76**	98.56

6.2 Approach 2: Models without Sensitive Features

In this approach, we remove sensitive features (sex and race) from the training data and evaluate fairness using a proxy-based counterfactual. Specifically, we use `relationship_Husband` as a proxy for sex: flipping this feature simulates a change in the individual's sex while maintaining semantic consistency with other relationship columns.
We compare three models:

- **Random Forest (RF)** — Standard model trained without sensitive features
- **MLP** — Neural network trained without sensitive features
- **SenSeI** — IBM's Sensitive Set Invariance method, which learns a distance metric to encourage invariance to sensitive attribute directions, even when those attributes are not directly available

6.2.1 Pareto Front Analysis

Figure 9 shows the Pareto fronts for all three models. Unlike Approach 1 where sensitive features were available, here we evaluate consistency by flipping the proxy feature (`relationship_Husband`).
The three models exhibit distinct trade-off profiles. SenSeI achieves the highest consistency (99.93% at best-accuracy knee-optimal) but the lowest accuracy (74.29%), reflecting its explicit fairness regularization that prioritizes invariance over predictive performance. RF occupies the middle ground with balanced performance (77.40% best accuracy, up to 100% consistency). MLP achieves the highest accuracy (80.57%) but the lowest consistency (90.57%), and notably shows the widest spread of configurations, indicating greater sensitivity to hyperparameter choices.
Despite removing sensitive features from the training data (compared to appraoch 1), all three models still exhibit some inconsistency, confirming that proxy features can leak sensitive information.

6.2.2 Hyperparameter Analysis

Figures 10 shows the parallel coordinate plots for each model in Approach 2. Bold lines indicate Pareto-optimal configurations, while faint lines represent dominated solutions.
For **Random Forest** (Figure 10a), Pareto-optimal configurations show strong concentration on `gini` criterion and `sqrt` for `max_features`, with notably no optimal configurations using all features (`None`).
For **MLP** (Figure 10b), well-performing configurations (dark green, low error) concentrate on `relu` or `tanh` activation with the `adam` solver and lower learning rates. Notably, the red lines

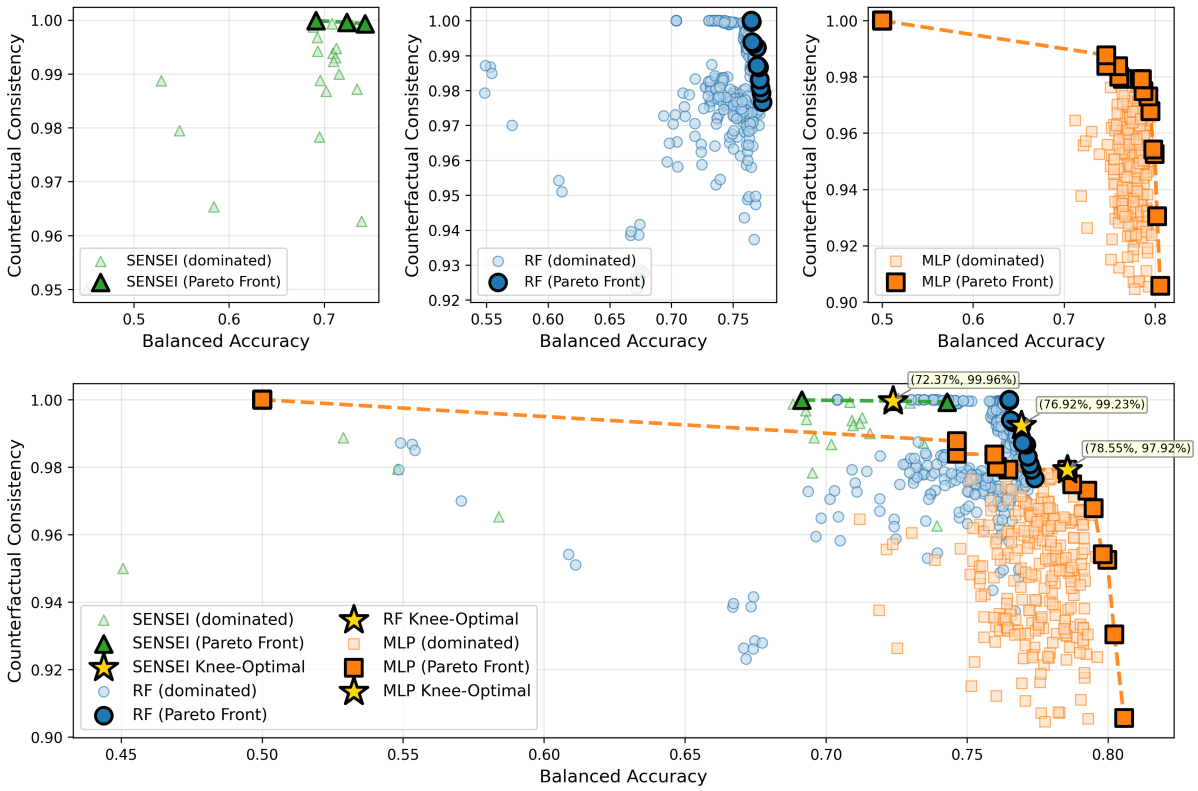
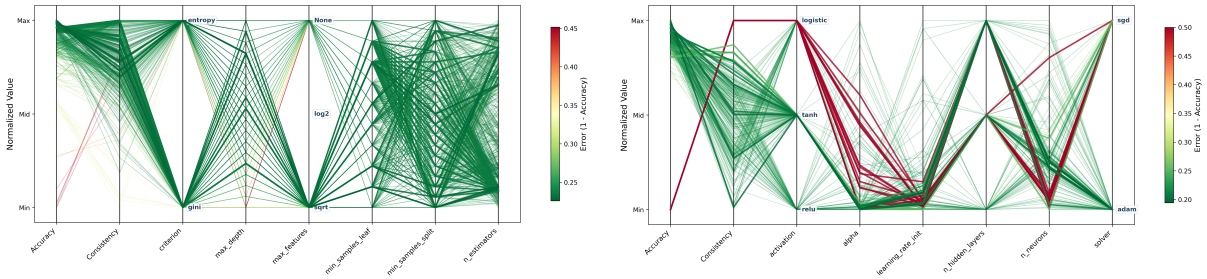
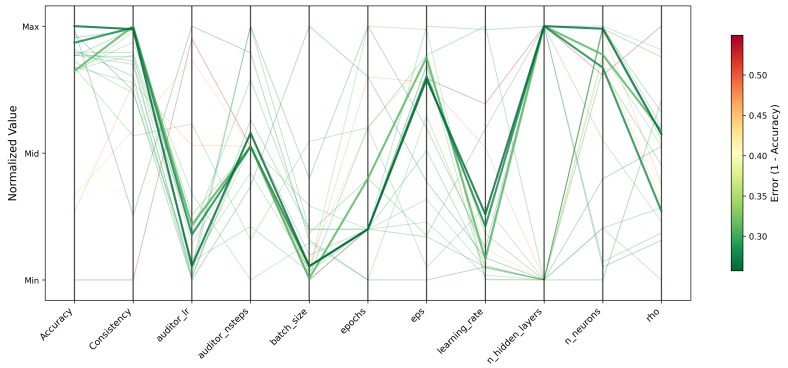


Figure 9: Pareto front for Approach 2 (proxy-based evaluation). All three models—RF, MLP, and SenSeI—are compared on the accuracy-consistency trade-off. Stars indicate knee-optimal points with their (accuracy, consistency) values.



(a) Random Forest

(b) MLP



(c) SenSeI

Figure 10: Parallel coordinate plots for Approach 2 showing hyperparameter-to-objective mappings.

representing high error are also Pareto-optimal—these correspond to *trivially fair* configurations that achieve perfect consistency but only random-level accuracy (see Section 6.2.3). These degenerate solutions consistently use `logistic` activation with `sgd` solver.
For **SenSeI** (Figure 10c), optimal configurations favor lower learning rates (both `learning_rate` and `auditor_lr`), smaller batch sizes, more training epochs, and larger network capacity (higher `n_hidden_layers` and `n_neurons`).

6.2.3 The Trivial Fairness Problem

During optimization, certain MLP configurations achieve perfect consistency (100%) but only approximately 50% accuracy—equivalent to random guessing. These represent *degenerate solutions* where the model has collapsed to a constant predictor rather than learning genuinely fair decision boundaries.
A constant predictor $f(x) = c$ trivially satisfies perfect counterfactual consistency:

$$\text{Consistency} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[f(x_i) = f(x_i')] = 1.0 \quad \text{when } f(x) = c \text{ for all } x \tag{10}$$

Root Cause: Vanishing Gradients. The degenerate configurations consistently use `logistic` (sigmoid) activation combined with `sgd` solver and high L2 regularization. The sigmoid function has a maximum gradient of only 0.25:

$$\sigma(x) = \frac{1}{1+e^{-x}}, \quad \sigma'(x) = \sigma(x)(1-\sigma(x)) \leq 0.25 \tag{11}$$

In a multi-layer network, gradients compound multiplicatively. For a 2-layer network:

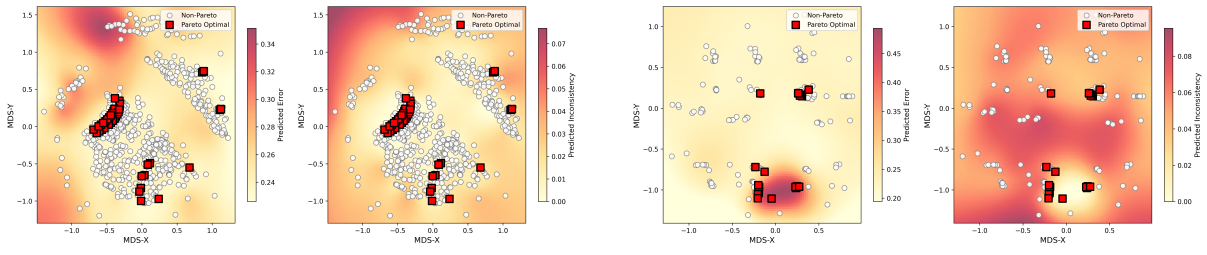
$$\text{Effective gradient} \approx 0.25 \times 0.25 = 0.0625 \tag{12}$$

Combined with SGD (no momentum to escape flat regions) and high regularization pushing weights toward zero, the network fails to learn and outputs near-constant predictions.

Implication for Fairness Evaluation. This phenomenon highlights an important caveat: *high fairness metrics alone do not guarantee a useful model*. When evaluating fairness-accuracy trade-offs, practitioners should verify that high-consistency models maintain meaningful predictive performance. The knee-optimal selection method naturally avoids these degenerate solutions by balancing both objectives.

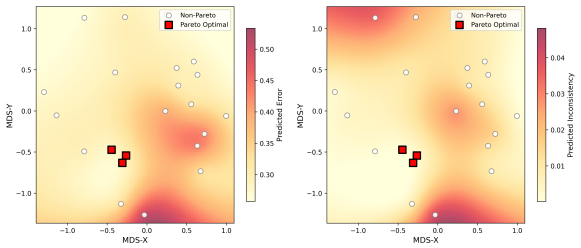
6.2.4 Configuration Space Visualization

Figure 11 presents MDS projections of the hyperparameter search space for each model.
Pareto-optimal configurations (red squares) tend to cluster together, indicating that optimal hyperparameter regions are localized rather than scattered throughout the search space. For RF (Figure 11a) and SenSeI (Figure 11c), Pareto-optimal points reside in yellow regions in both panels, meaning these configurations achieve low error *and* low inconsistency simultaneously—no trade-off is required.
In contrast, MLP (Figure 11b) exhibits behavior similar to Approach 1: Pareto-optimal points appear in multiple distinct clusters, with some located in higher-error (darker) regions of the left panel. These correspond to the trivially fair configurations discussed in Section 6.2.3, which achieve perfect consistency at the cost of degraded accuracy. This visual separation reinforces that MLP's Pareto front spans from high-accuracy/lower-consistency solutions to degenerate constant predictors.



(a) RF

(b) MLP



(c) SenSeI

Figure 11: MDS projection of the hyperparameter space for Approach 2.

6.2.5 Test Set Evaluation

Table 5 summarizes the performance of knee-optimal configurations on the held-out test set, and Figure 12 visualizes the accuracy-consistency trade-off.

Table 5: Test set performance for Approach 2 (knee-optimal configurations). Best values per metric in bold.

Model	Accuracy (%)	Balanced Acc. (%)	Consistency (%)
RF	84.36	75.45	99.16
MLP	**85.01**	**77.62**	97.6
SenSeI	72.37	72.37	**99.96**

The results reveal distinct trade-off profiles for each model type. **MLP** achieves the highest accuracy (85.01%) and balanced accuracy (77.62%), but at the cost of lower consistency (97.60%). **RF** provides a balanced middle ground with strong accuracy (84.36%) and high consistency (99.16%). **SenSeI**, the fairness-aware training method, achieves the highest consistency (99.96%) but sacrifices approximately 12–13 percentage points in accuracy compared to RF and MLP. This hierarchy—MLP prioritizing accuracy, SenSeI prioritizing fairness, and RF balancing both—aligns with each model's design philosophy. Notably, RF achieves consistency comparable to SenSeI (99.16% vs 99.96%) while maintaining accuracy competitive with MLP, suggesting that careful hyperparameter optimization can yield fairness benefits without requiring specialized fairness-aware training algorithms.

7 Conclusion

This work demonstrates that framing fairness-aware model selection as a multi-objective hyperparameter optimization problem enables systematic exploration of the accuracy–fairness

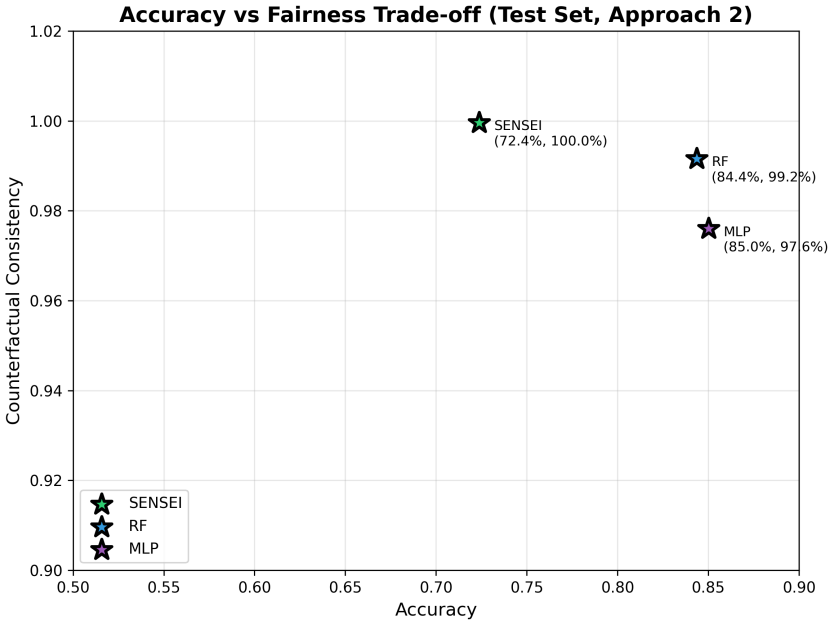


Figure 12: Accuracy vs fairness trade-off for Approach 2 on test set. Each star represents a knee-optimal model.

trade-off landscape. By leveraging AutoML techniques, specifically SMAC with ParEGO, we automatically identify diverse Pareto-optimal configurations that represent different trade-offs between predictive accuracy and counterfactual consistency. Our experiments on the Adult Income dataset reveal that significant fairness improvements can be achieved with relatively small accuracy sacrifices, and that different model types exhibit distinct trade-off characteristics: Random Forest provides balanced performance, MLP prioritizes accuracy, and SenSeI prioritizes fairness. The knee-optimal selection method offers a principled approach for practitioners to select models that balance both objectives, while the Pareto front visualization provides transparency into the available trade-offs. These findings suggest that automated hyperparameter optimization is a valuable tool for fairness-aware machine learning, enabling practitioners to make informed decisions about model selection based on their specific accuracy and fairness requirements.

References

[1] Solon Barocas, Moritz Hardt, and Arvind Narayanan. *Fairness and Machine Learning*. fairmlbook.org, 2019.

[2] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 214–226, 2012.

[3] IBM Research. inFairness Library. https://ibm.github.io/inFairness/, 2023.

[4] Mikhail Yurochkin and Yuekai Sun. SenSeI: Sensitive Set Invariance for Enforcing Individual Fairness. In *International Conference on Learning Representations (ICLR)*, 2021.

[5] Matt J. Kusner, Joshua Loftus, Chris Russell, and Ricardo Silva. Counterfactual fairness. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[6] Aditya Krishna Menon and Robert C. Williamson. The cost of fairness in binary classification. In *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, volume 81, pages 107–118, 2018.

[7] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[8] Barry Becker and Ronny Kohavi. Adult [Dataset]. UCI Machine Learning Repository, 1996. https://archive.ics.uci.edu/dataset/2/adult. DOI: 10.24432/C5XW20.

[9] Joshua Knowles. ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10(1):50–66, 2006.

[10] Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, René Sass, and Frank Hutter. SMAC3: A versatile Bayesian optimization package for hyperparameter optimization. *Journal of Machine Learning Research*, 23(54):1–9, 2022.

[11] Marius Lindauer, Katharina Eggensperger, Matthias Feurer, Stefan Falkner, André Biedenkapp, and Frank Hutter. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems*, volume 28, 2015.