ICML 2025 Rebuttal

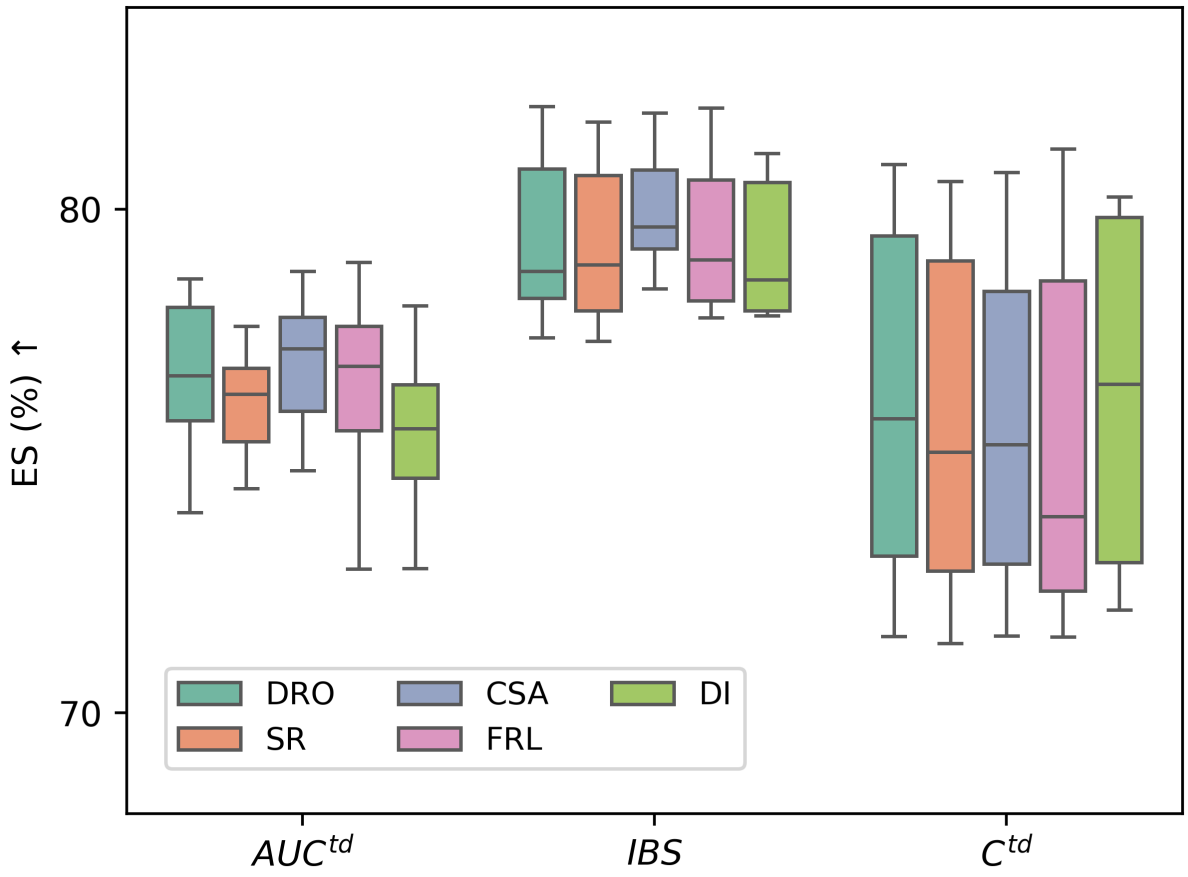

Figure R1: Fairness-utility trade-offs of fairness algorithms for TTE prediction across various utility metrics under distribution shift created by flipping censoring indices (shift on Δ). For each metric, we compute the corresponding equity scaling score as a measure of the trade-off. The results for each fairness algorithm are aggregated across all dataset and sensitive attribute combinations.

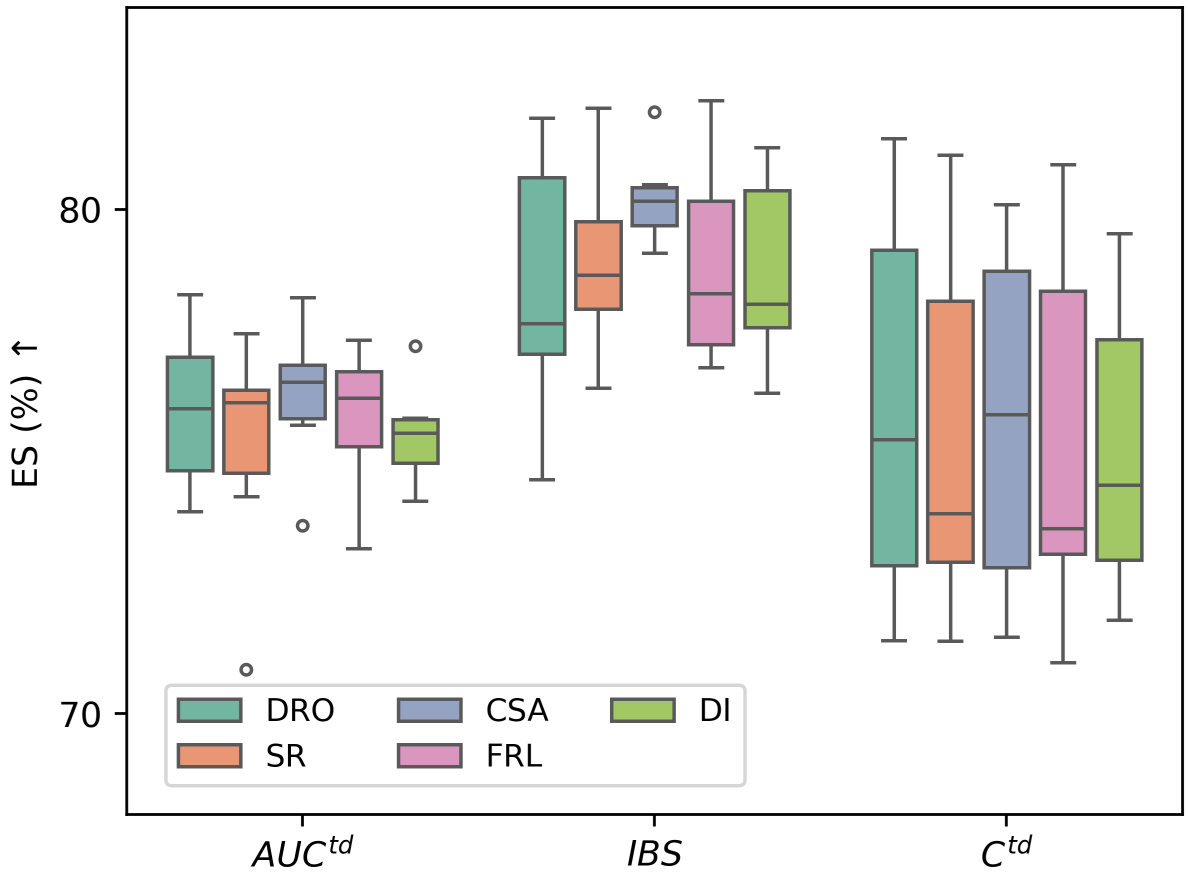

Figure R2: Fairness-utility trade-offs of fairness algorithms for TTE prediction across various utility metrics under distribution shift created by adding noise to images (shift on X). For each metric, we compute the corresponding equity scaling score as a measure of the trade-off. The results for each fairness algorithm are aggregated across all dataset and sensitive attribute combinations.

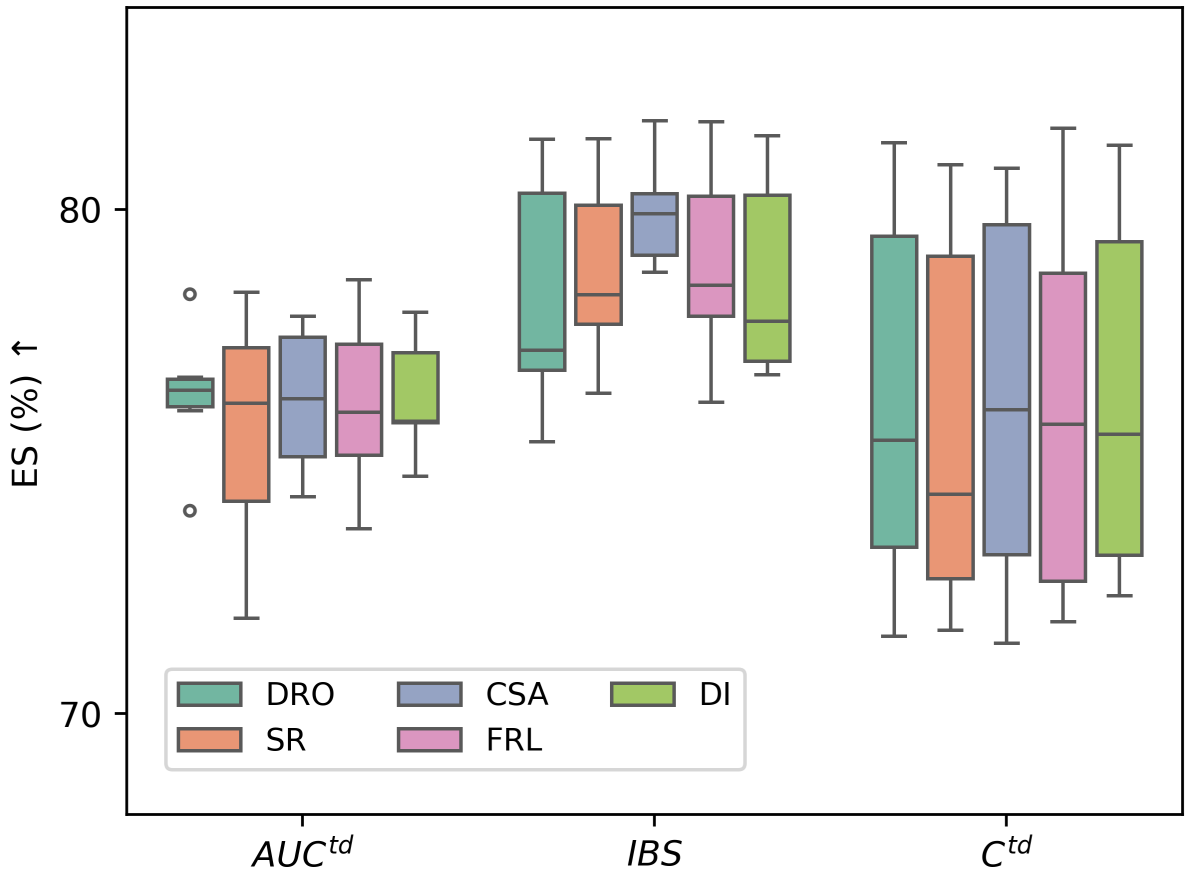

Figure R3: Fairness-utility trade-offs of fairness algorithms for TTE prediction across various utility metrics under distribution shift created by adding noise to TTE labels (shift on Y). For each metric, we compute the corresponding equity scaling score as a measure of the trade-off. The results for each fairness algorithm are aggregated across all dataset and sensitive attribute combinations.

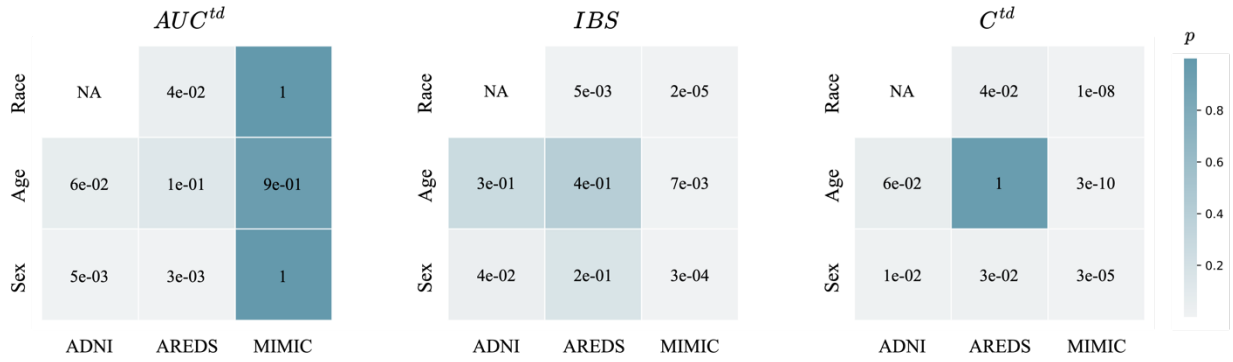

Figure R4: P-values from the one-sided Wilcoxon signed-rank test computed across all fair TTE prediction models and random seeds. A p-value < 0.05 suggests distribution shift on Y significantly degrades TTE predictive performance compared no distribution shift.

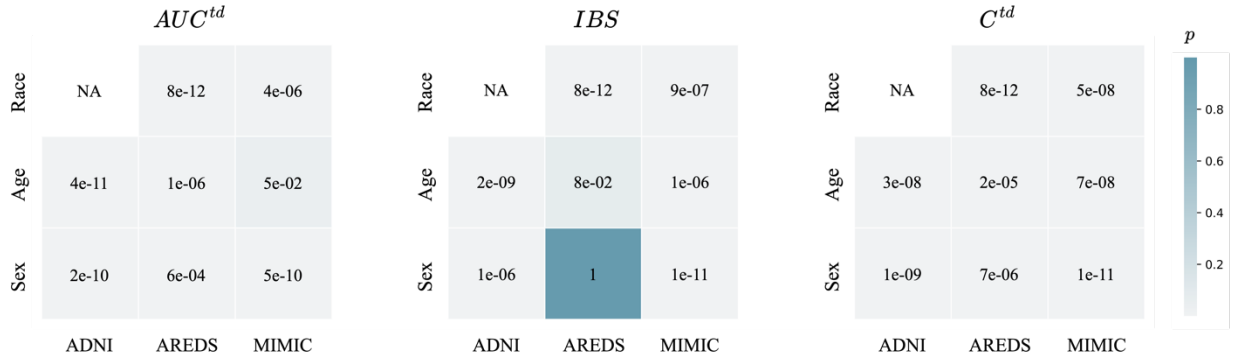

Figure R5: P-values from the one-sided Wilcoxon signed-rank test computed across all fair TTE prediction models and random seeds. A p-value < 0.05 suggests distribution shift on X significantly degrades TTE predictive performance compared to no distribution shift.

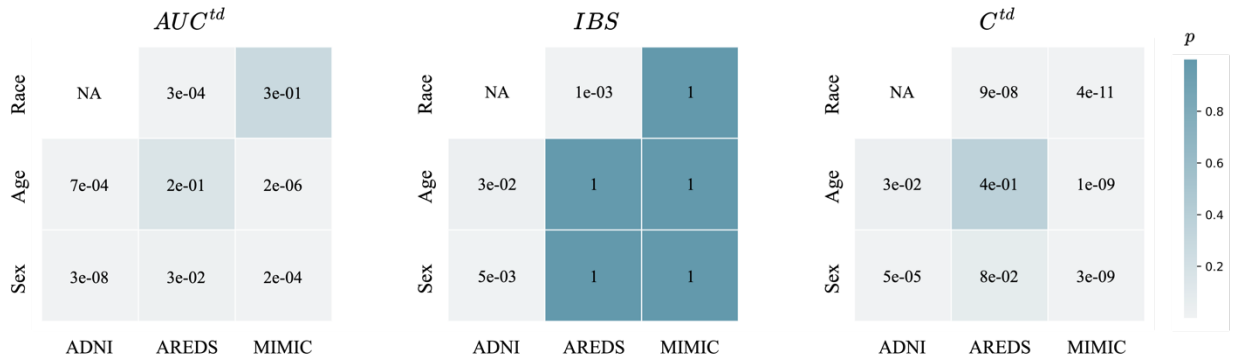

Figure R6: P-values from the one-sided Wilcoxon signed-rank test computed across all fair TTE prediction models and random seeds. A p-value < 0.05 suggests distribution shift on Δ significantly degrades TTE predictive performance compared to no distribution shift.

```python
def discretize_samples(x, y, bins=10):
    """
    Discretize two continuous samples into bins.
    Returns joint and marginal distributions.
    """
    # Discretize the two samples
    hist_2d, x_edges, y_edges = np.histogram2d(x, y, bins=bins)

    # Normalize to get probabilities
    joint_prob = hist_2d / hist_2d.sum()
    marginal_x = np.sum(joint_prob, axis=1)
    marginal_y = np.sum(joint_prob, axis=0)

    return joint_prob, marginal_x, marginal_y

def calculate_nmi(x, y, bins=10):
    """
    Calculate the normalized mutual information between two continuous samples.
    """
    # Discretize the samples
    joint_prob, marginal_x, marginal_y = discretize_samples(x, y, bins=bins)

    # Flatten joint probability for mutual information calculations
    joint_flat = joint_prob[joint_prob > 0]

    # Compute mutual information
    mutual_info = np.sum(joint_flat * np.log(joint_flat / (np.outer(marginal_x, marginal_y))[joint_prob > 0].flatten()))

    # Compute entropy for normalization
    entropy_x = -np.sum(marginal_x[marginal_x > 0] * np.log(marginal_x[marginal_x > 0]))
    entropy_y = -np.sum(marginal_y[marginal_y > 0] * np.log(marginal_y[marginal_y > 0]))

    # Normalize mutual information
    nmi = mutual_info / np.sqrt(entropy_x * entropy_y)
    return nmi
```

Figure R7: Python code to compute normalized mutual information between two random variables. This code is used to quantify sources of bias including disparity in mutual information between X_Z and Y, and disparity in mutual information between X_Z and Δ.