Holistic AI × The Alan Turing Institute

Bias in Regression Tasks – Part IV

Content by: Sachin Beepath, Giulio Filippi, Nigel Kingsman, Cristian Munoz, Roseline Polle, Sara Zannone

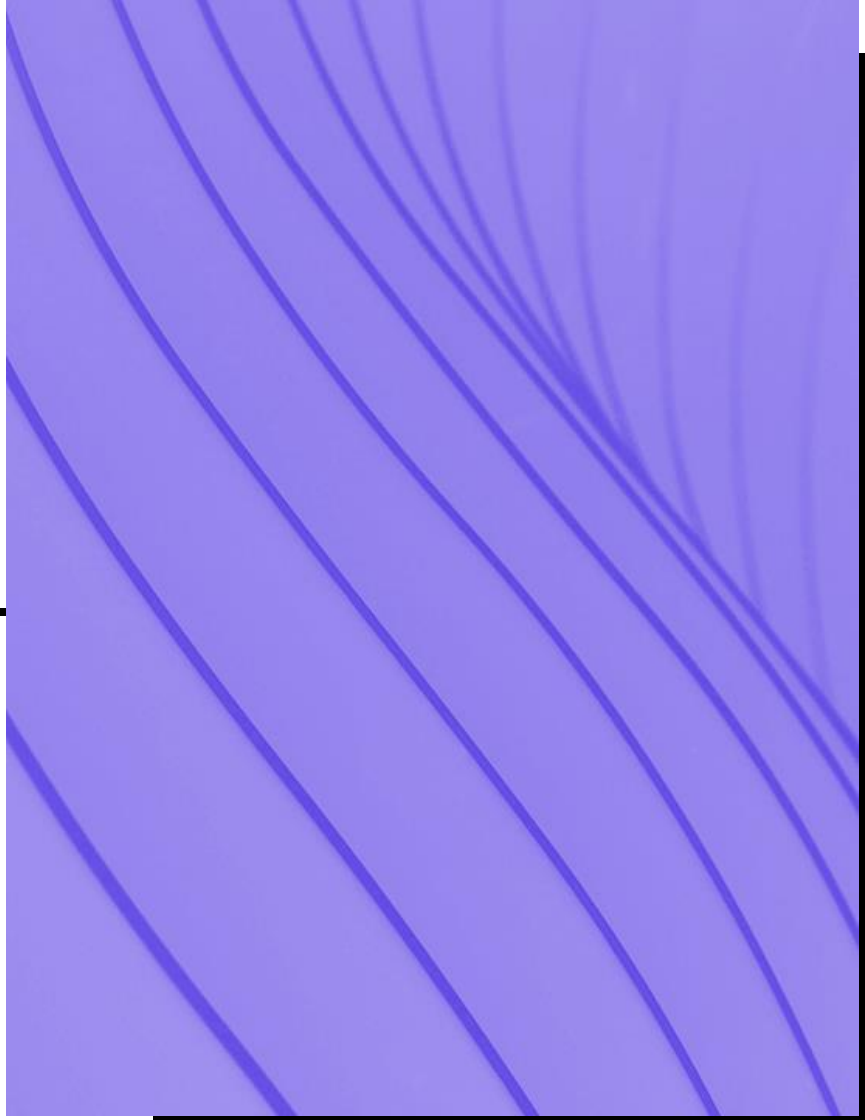
Speaker: Sara Zannone

Contents

- Part I – Introduction to Regression
- Part II – Fairness in Regression
- Part III – Measuring Bias in Regression
- **Part IV – Mitigating Bias in Regression**

Mitigating Bias in Regression

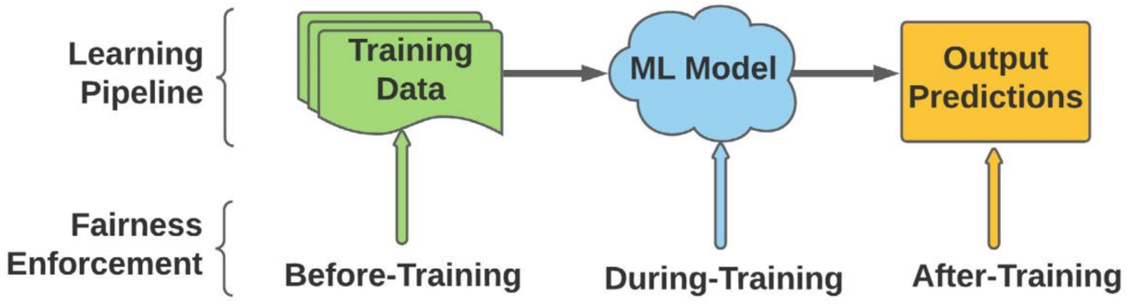
1. Introduce different levels of bias mitigation.
2. Present techniques from different levels.
3. Implement mitigation techniques with the holisticai library



Types of Mitigation Techniques

- **Pre-Processing**

Occurs ***before*** training by modifying the original dataset. This ensures that the model outputs meet the fairness requirements.



- **In-Processing**

Occurs ***during*** training. The model or learning process is changed to ensure that the outputs will meet the fairness requirements.

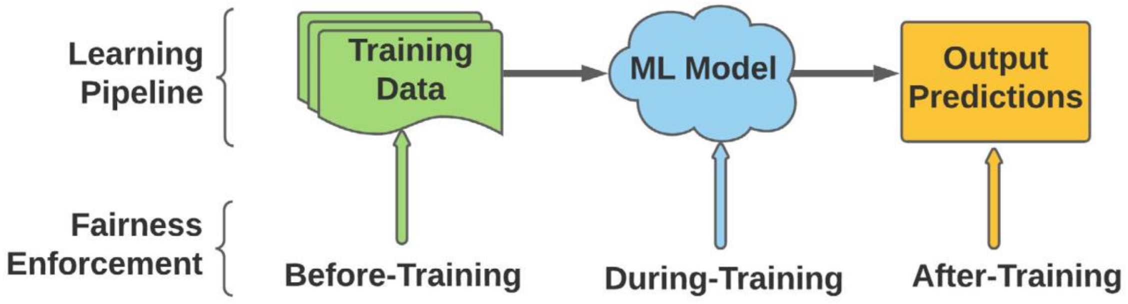
- **Post-Processing**

Occurs ***after*** training. The outputs of the model are modified to meet the fairness requirements.

Types of Mitigation Techniques

- **Pre-Processing**

Occurs ***before*** training by modifying the original dataset. This ensures that the model outputs meet the fairness requirements.



- **In-Processing**

Occurs ***during*** training. The model or learning process is changed to ensure that the outputs will meet the fairness requirements.

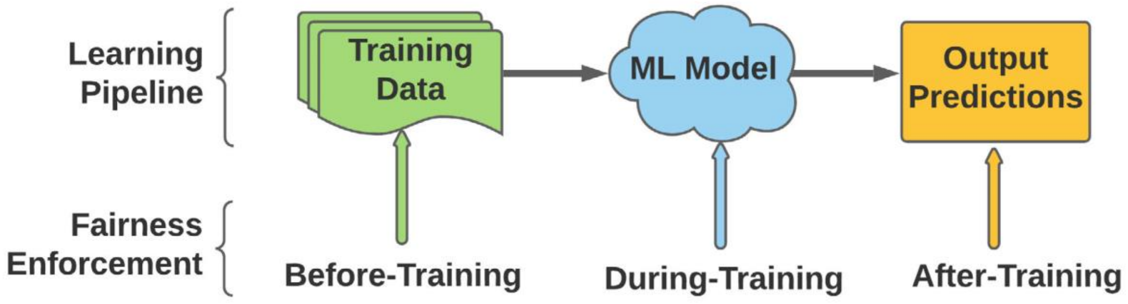
- **Post-Processing**

Occurs ***after*** training. The outputs of the model are modified to meet the fairness requirements.

Types of Mitigation Techniques

- **Pre-Processing**

Occurs ***before*** training by modifying the original dataset. This ensures that the model outputs meet the fairness requirements.



- **In-Processing**

Occurs ***during*** training. The model or learning process is changed to ensure that the outputs will meet the fairness requirements.

- **Post-Processing**

Occurs ***after*** training. The outputs of the model are modified to meet the fairness requirements.

Pre-processing Bias Mitigation

- Occurs before learning and makes changes to the training dataset.
- It works with any model (model-agnostic)
- Original dataset X is transformed to X'.
- The algorithm remains the same but removing the bias from the dataset helps reducing bias in the outputs.

Pre-processing Bias Mitigation

- Occurs before learning and makes changes to the training dataset.
- It works with any model (model-agnostic)
- Original dataset X is transformed to X'.
- The algorithm remains the same but removing the bias from the dataset helps reducing bias in the outputs.

Correlation Remover

- Pre-processing technique

- It is well known that removing protected attributes from the dataset (Fairness through unawareness) is not sufficient to prevent biased outputs. This is because of proxy variables in the data.

- One obvious solution is to reduce the correlation between protected attributes and the other variables

Correlation Remover

- Pre-processing technique
- It is well known that removing protected attributes from the dataset (Fairness through unawareness) is not sufficient to prevent biased outputs. This is because of proxy variables in the data.
- One obvious solution is to reduce the correlation between protected attributes and the other variables

Correlation Remover

- Pre-processing technique
- It is well known that removing protected attributes from the dataset (Fairness through unawareness) is not sufficient to prevent biased outputs. This is because of proxy variables in the data.
- One obvious solution is to reduce the correlation between protected attributes and the other variables

Correlation Remover

- Goal:
  - De-correlate the non-sensitive features from the protected attributes
  - While retaining as much information as possible
- Mathematically, this is achieved by:
  - Applying a linear transformation to the non-sensitive feature columns that essentially projects away their correlation with protected attributes.
  - If *X* is the original dataset, *Z* are the non-sensitive features and *S* is the set of protected attributes, then we'll have that:

$$\min_{\mathbf{z}_1,\ldots,\mathbf{z}_n} \sum_{i=1}^{n} \|\mathbf{z}_i - \mathbf{x}_i\|^2$$

$$\text{subject to}$$

$$\frac{1}{n}\sum_{i=1}^{n} \mathbf{z}_i(\mathbf{s}_i - \overline{\mathbf{s}})^T = \mathbf{0}$$

Correlation Remover

- Goal:
  - De-correlate the non-sensitive features from the protected attributes
  - While retaining as much information as possible
- Mathematically, this is achieved by:
  - Applying a linear transformation to the non-sensitive feature columns that essentially projects away their correlation with protected attributes.
  - If *X* is the original dataset, *Z* are the non-sensitive features and *S* is the set of protected attributes, then we'll have that:

$$\min_{\mathbf{z}_1,\ldots,\mathbf{z}_n} \sum_{i=1}^{n} \|\mathbf{z}_i - \mathbf{x}_i\|^2$$

subject to

$$\frac{1}{n}\sum_{i=1}^{n} \mathbf{z}_i(\mathbf{s}_i - \overline{\mathbf{s}})^T = \mathbf{0}$$

Correlation Remover

- Goal:
  - De-correlate the non-sensitive features from the protected attributes
  - While retaining as much information as possible
- Mathematically, this is achieved by:
  - Applying a linear transformation to the non-sensitive feature columns that essentially projects away their correlation with protected attributes.
  - If *X* is the original dataset, *Z* are the non-sensitive features and *S* is the set of protected attributes, then we'll have that:

$$\min_{\mathbf{z}_1,\ldots,\mathbf{z}_n} \sum_{i=1}^{n} \|\mathbf{z}_i - \mathbf{x}_i\|^2$$

$$\text{subject to}$$

$$\frac{1}{n}\sum_{i=1}^{n} \mathbf{z}_i(\mathbf{s}_i - \overline{\mathbf{s}})^T = \mathbf{0}$$

Correlation Remover

- Goal:
  - de-correlate the non-sensitive features from the protected attributes
  - while retaining as much information as possible

- This method changes the original dataset by removing correlation with protected attributes.

- Note that the correlation measures linear relationships, so it might still be possible that features are dependent on protected attributes in a non-linear way.

Correlation Remover with holisticai library

- The first step is installing the library

```
# install the holisticai library
!pip install holisticai
```

- We can now import the Correlation Remover mitigation technique

```
from holisticai.bias.mitigation import CorrelationRemover
```

- We then initialise our chosen model and create the training pipeline

```
model = LinearRegression()
pipeline = Pipeline(
    steps=[
        ('scalar', StandardScaler()),
        ("bm_preprocessing", CorrelationRemover()),
        ("model", model),
    ]
)
```

Correlation Remover with holisticai library

- The first step is installing the library

```
# install the holisticai library
!pip install holisticai
```

- We can now import the Correlation Remover mitigation technique

```
from holisticai.bias.mitigation import CorrelationRemover
```

- We then initialise our chosen model and create the training pipeline

```
model = LinearRegression()
pipeline = Pipeline(
    steps=[
        ('scalar', StandardScaler()),
        ("bm_preprocessing", CorrelationRemover()),
        ("model", model),
    ]
)
```

Correlation Remover with holisticai library

- The first step is installing the library

```
# install the holisticai library
!pip install holisticai
```

- We can now import the Correlation Remover mitigation technique

```
from holisticai.bias.mitigation import CorrelationRemover
```

- We then initialise our chosen model and create the training pipeline

```
model = LinearRegression()
pipeline = Pipeline(
    steps=[
        ('scalar', StandardScaler()),
        ("bm_preprocessing", CorrelationRemover()),
        ("model", model),
    ]
)
```

Correlation Remover with holisticai library

- After creating the group membership vectors, we can fit the pipeline

```
X, y, group_a, group_b = train_data
fit_params = {
    "bm__group_a": group_a,
    "bm__group_b": group_b
}

pipeline.fit(X, y, **fit_params)
```

- Finally, we can test our pipeline on the test data

```
X, y, group_a, group_b = test_data
predict_params = {
    "bm__group_a": group_a,
    "bm__group_b": group_b,
}
y_pred = pipeline.predict(X, **predict_params)
```

Correlation Remover with holisticai library

- After creating the group membership vectors, we can fit the pipeline

```
X, y, group_a, group_b = train_data
fit_params = {
    "bm__group_a": group_a,
    "bm__group_b": group_b
}

pipeline.fit(X, y, **fit_params)
```

- Finally, we can test our pipeline on the test data

```
X, y, group_a, group_b = test_data
predict_params = {
    "bm__group_a": group_a,
    "bm__group_b": group_b,
}
y_pred = pipeline.predict(X, **predict_params)
```

In-processing Bias Mitigation

- Occurs during learning.
- Makes changes to the algorithm, usually on the optimization part (model-specific).
- Trade-off between fairness and accuracy
- The dataset is left unchanged

In-processing Bias Mitigation

- Occurs during learning.
- Makes changes to the algorithm, usually on the optimization part (model-specific).
- Trade-off between fairness and accuracy
- The dataset is left unchanged

Exponentiated Gradient Reduction

- In-processing technique, can be used for both classification (Agarwal et al., 2018) and regression (Agarwal et al., 2019)

- Fair regression aims to minimize the expected loss while guaranteeing a fairness constraint

- If we consider Bounded Group Loss, we want to find the function f such that :

$$\min_{f \in F} \mathbb{E}[\ell(y, f(\boldsymbol{x}))] \quad \text{such that } \forall a \in A$$

$$\mathbb{E}[\ell(y, f(\boldsymbol{x}))|A = a] \leq \eta$$

Exponentiated Gradient Reduction

- In-processing technique, can be used for both classification (Agarwal et al., 2018) and regression (Agarwal et al., 2019)

- Fair regression aims to minimize the expected loss while guaranteeing a fairness constraint

- If we consider Bounded Group Loss, we want to find the function f such that :

$$\min_{f \in F} \mathbb{E}[\ell(y, f(\boldsymbol{x}))] \quad \text{such that } \forall a \in A$$

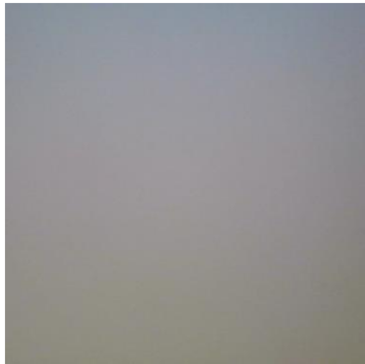
$$\mathbb{E}[\ell(y, f(\boldsymbol{x}))|A = a] \leq \eta$$

Exponentiated Gradient Reduction

- Exponentiated Gradient Reduction works by selecting randomized predictors, which:
  - first pick f according to a distribution Q
  - then predict according to f

$$
\min_{Q \in \Delta(F)} \sum_{f} \mathrm{Q}(f) \cdot \mathbb{E}[\ell(y, f(\boldsymbol{x}))] \quad \text{such that } \forall a \in A
$$

$$
\sum_{f} \mathrm{Q}(f) \cdot \mathbb{E}[\ell(y, f(\boldsymbol{x})) | A = a] \leq \eta
$$



Exponentiated Gradient Reduction

- Exponentiated Gradient Reduction works by selecting randomized predictors, which:
  - first pick f according to a distribution Q
  - then predict according to f

$$\min_{Q \in \Delta(F)} \sum_f \mathrm{Q}(f) \cdot \mathbb{E}[\ell(y, f(\boldsymbol{x}))] \quad \text{such that } \forall a \in A$$

$$\sum_f \mathrm{Q}(f) \cdot \mathbb{E}[\ell(y, f(\boldsymbol{x}))|A = a] \leq \eta$$

Exponentiated Gradient Reduction

$$\min_{Q \in \Delta(F)} \widehat{loss(Q)} \text{ such that } \widehat{\gamma_{BGL}(Q)} \leq \eta \quad \forall a \in A$$

- This can then be then transformed into a Lagrangian and solved as an optimization problem:

$$L^{BGL}(Q,\lambda) = \widehat{loss(Q)} + \sum_{a} \lambda_a(\gamma_{BGL}(Q) - \eta)$$

- The goal is to find the saddle point, which is guaranteed to exist.

Post-processing Bias Mitigation

- Occurs after learning.
- Makes changes to the outputs directly.
- The training dataset and the algorithm remain the same.
- Model-agnostic.

Wasserstein Barycenters for Fair Regression

- Post-processing technique for regression (Chzen et al., 2020; Le Gouic 2020)

- The Wasserstein barycenter problem is well-known in optimal transport theory.

- Wasserstein barycenters provide a natural approach for averaging probability distributions in a way that respects their geometry

Wasserstein Barycenters for Fair Regression

This technique is based on the following property:

- **The distribution of the optimal fair predictor is the solution of a Wasserstein barycenter problem between the distributions induced by the unfair regression function on the sensitive groups.**

Wasserstein Barycenters for Fair Regression

This technique is based on the following property:

- **The distribution of the optimal fair predictor is the solution of a Wasserstein barycenter problem between the distributions induced by the unfair regression function on the sensitive groups.**

Wasserstein Barycenters - example

- Let’s consider an example with binary protected attributes.

- Candidates belong to group 1 and group 2 with probabilities respectively: $p_1 = 2/5$ and $p_2 = 3/5$

- x = candidate’s CV
- s = candidate’s group
- $f^*(x, s)$ = current market’s salary (unfair)
- $t^*(x, s)$ = adjusted salary

Wasserstein Barycenters - example

- Let’s consider an example with binary protected attributes.

- Candidates belong to group 1 and group 2 with probabilities respectively: $p_1 = 2/5$ and $p_2 = 3/5$

- x = candidate’s CV
- s = candidate’s group
- $f^*(x, s)$ = current market’s salary (unfair)
- $t^*(x, s)$ = adjusted salary

Wasserstein Barycenters - example

Let's consider a candidate x from group 1. The current market's salary will then be: $f^*(x, 1)$. How to compute the adjusted salary:

1. Compute the fraction of individuals from the first group whose market salary is at most $f^*(x, 1)$

Wasserstein Barycenters - example

Let's consider a candidate x from group 1. The current market's salary will then be: $f^*(x,1)$. How to compute the adjusted salary:

1. Compute the fraction of individuals from the first group whose market salary is at most $f^*(x,1)$
2. Find a candidate $\bar{x}$ in group 2, such that the fraction of individuals from the second group whose market salary is at most $f^*\ (\bar{x},2)$ is the same:
   $P(f^*(X,S) \leq f^*(x,1)|S=1) = \ P(f^*(X,S) \leq f^*(\bar{x},2)|S=2)$

Wasserstein Barycenters - example

Let's consider a candidate x from group 1. The current market's salary will then be: $f^*(x, 1)$. How to compute the adjusted salary:

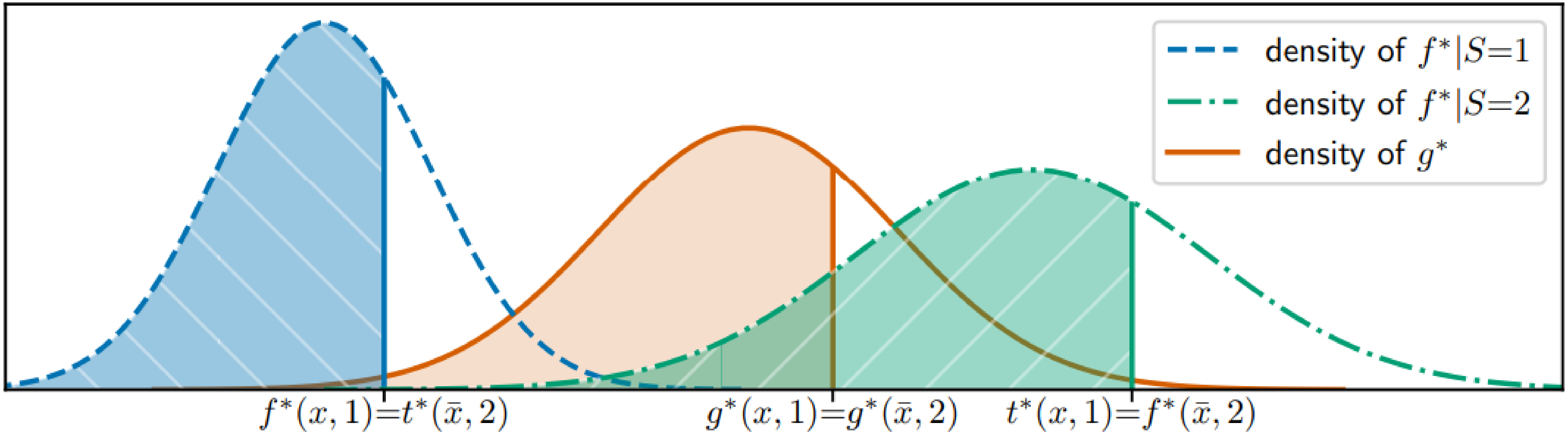
1. Compute the fraction of individuals from the first group whose market salary is at most $f^*(x, 1)$
2. Find a candidate $\bar{x}$ in group 2, such that the fraction of individuals from the second group whose market salary is at most $f^*\ (\bar{x}, 2)$ is the same:
   $P(f^*(X, S) \leq f^*(x, 1) | S = 1) = \ P(f^*(X, S) \leq f^*(\bar{x}, 2) | S = 2)$
3. The market salary of $\bar{x}$ is exactly the adjustment for x:
   $$t^*(x, 1) = f^*\ (\bar{x}, 2)$$

- If candidates $(x, 1)$ and $(\bar{x}, 2)$ have the same market salary ranking in their group, then they should receive the same salary

- The fair salary is determined by:

$$g^*(x, 1) = g^*(\bar{x}, 2) = p_1 f^*(x, 1) + p_2 f^*(\bar{x}, 2)$$

Fair optimal prediction g^* with $p_1 = 2/5$ and $p_2 = 3/5$

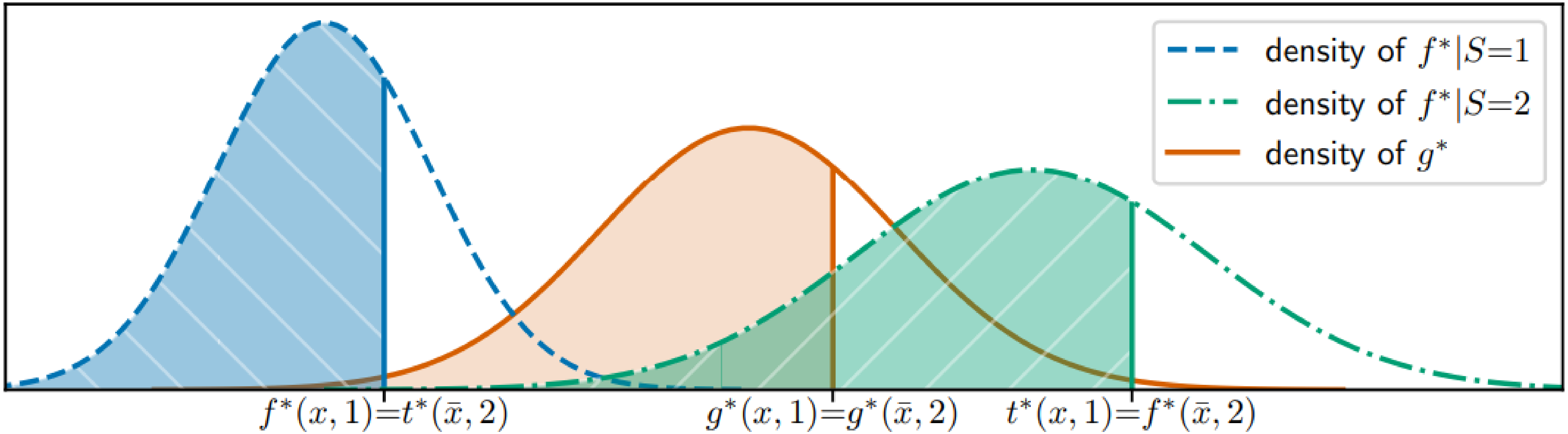


- If candidates $(x, 1)$ and $(\bar{x}, 2)$ have the same market salary ranking in their group, then they should receive the same salary

- The fair salary is determined by:

$$g^*(x,1) = g^*(\bar{x},2) = p_1 f^*(x,1) + p_2 f^*(\bar{x},2)$$

Fair optimal prediction g^* with $p_1 = 2/5$ and $p_2 = 3/5$



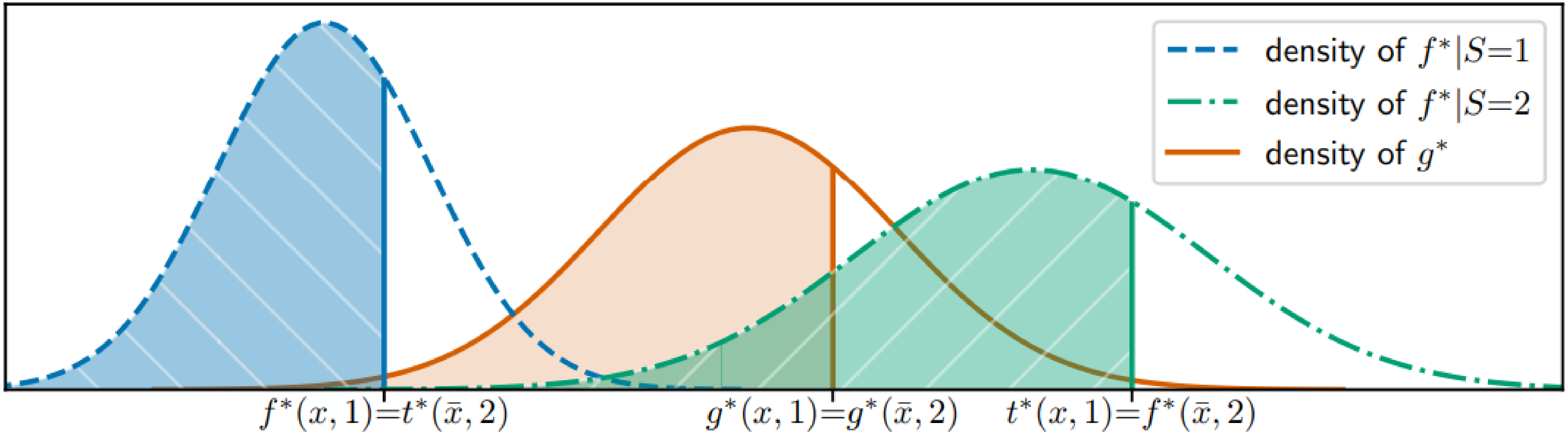
- The fair salary is determined by:

$$g^*(x,1) = g^*(\bar{x},2) = p_1 f^*(x,1) + p_2 f^*(\bar{x},2)$$

- The difference in salary for a fair decision:

$$\Delta(p_2 - p_1)\big(f^*(\bar{x},2) - f^*(\bar{x},1)\big)$$

Fair optimal prediction g^* with $p_1 = 2/5$ and $p_2 = 3/5$



References and Links

[1] Correlation Remover, FairLearn, https://fairlearn.org/main/user_guide/mitigation/preprocessing.html

[2] Agarwal, Alekh, et al. "A reductions approach to fair classification." *International Conference on Machine Learning*. PMLR, 2018.

[3] Agarwal, Alekh, Miroslav Dudík, and Zhiwei Steven Wu. "Fair regression: Quantitative definitions and reduction-based algorithms." *International Conference on Machine Learning*. PMLR, 2019.

[4] Chzhen, Evgenii, et al. "Fair regression with wasserstein barycenters." *Advances in Neural Information Processing Systems* 33 (2020): 7321-7331.

[5] Gouic, Thibaut Le, Jean-Michel Loubes, and Philippe Rigollet. "Projection to fairness in statistical learning." *arXiv preprint arXiv:2005.11720* (2020).

[6] *Pipeline, scikit learn, https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html*

[7] Holisticai library documentation, https://holisticai.readthedocs.io/en/latest/