In [2]:

```python
# Load all necessary packages
import sys
sys.path.insert(1, "../")

import numpy as np
np.random.seed(0)

from aif360.datasets import GermanDataset
from aif360.metrics import BinaryLabelDatasetMetric, DatasetMetric
from aif360.algorithms.preprocessing import Reweighing
from aif360.explainers import MetricTextExplainer, MetricJSONExplainer

from IPython.display import Markdown, display

import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns

import json
from collections import OrderedDict
```

```
WARNING:root:
`load_boston` has been removed from scikit-learn since version 1.2.

The Boston housing prices dataset has an ethical problem: as
investigated in [1], the authors of this dataset engineered a
non-invertible variable "B" assuming that racial self-segregation had a
positive impact on house prices [2]. Furthermore the goal of the
research that led to the creation of this dataset was to study the
impact of air quality but it did not give adequate demonstration of the
validity of this assumption.

The scikit-learn maintainers therefore strongly discourage the use of
this dataset unless the purpose of the code is to study and educate
about ethical issues in data science and machine learning.

In this special case, you can fetch the dataset from the original
source::

    import pandas as pd
    import numpy as np

    data_url = "http://lib.stat.cmu.edu/datasets/boston"
    raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22, header=None)
    data = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :2]])
    target = raw_df.values[1::2, 2]

Alternative datasets include the California housing dataset and the
Ames housing dataset. You can load the datasets as follows::

    from sklearn.datasets import fetch_california_housing
    housing = fetch_california_housing()

for the California housing dataset and::

    from sklearn.datasets import fetch_openml
    housing = fetch_openml(name="house_prices", as_frame=True)

for the Ames housing dataset.

[1] M Carlisle.
"Racist data destruction?"
<https://medium.com/@docintangible/racist-data-destruction-113e3eff54a8>

[2] Harrison Jr, David, and Daniel L. Rubinfeld.
"Hedonic housing prices and the demand for clean air."
Journal of environmental economics and management 5.1 (1978): 81-102.
<https://www.researchgate.net/publication/4974606_Hedonic_housing_prices_and_the_demand_for_clean_air>
: LawSchoolGPADataset will be unavailable. To install, run:
pip install 'aif360[LawSchoolGPA]'
WARNING:tensorflow:From C:\Users\ramgo\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecated. Please use tf.co
mpat.v1.losses.sparse_softmax_cross_entropy instead.
```

In [3]:

```python
dataset_orig = GermanDataset(protected_attribute_names=['age'],           # this dataset also contains protected
                                                                          # attribute for "sex" which we do not
                                                                          # consider in this evaluation
                             privileged_classes=[lambda x: x >= 25],      # age >=25 is considered privileged
                             features_to_drop=['personal_status', 'sex']) # ignore sex-related attributes

dataset_orig_train, dataset_orig_test = dataset_orig.split([0.7], shuffle=True)

privileged_groups = [{'age': 1}]
unprivileged_groups = [{'age': 0}]
```

In [4]:

```python
print("Original one hot encoded german dataset shape: ",dataset_orig.features.shape)
print("Train dataset shape: ", dataset_orig_train.features.shape)
print("Test dataset shape: ", dataset_orig_test.features.shape)
```

```
Original one hot encoded german dataset shape:  (1000, 57)
Train dataset shape:  (700, 57)
Test dataset shape:  (300, 57)
```

In [5]:

```python
df, dict_df = dataset_orig.convert_to_dataframe()
```

In [6]:

```python
print("Shape: ", df.shape)
print(df.columns)
df.head(5)
```

```
Shape:  (1000, 58)
Index(['month', 'credit_amount', 'investment_as_income_percentage',
       'residence_since', 'age', 'number_of_credits', 'people_liable_for',
       'status=A11', 'status=A12', 'status=A13', 'status=A14',
       'credit_history=A30', 'credit_history=A31', 'credit_history=A32',
       'credit_history=A33', 'credit_history=A34', 'purpose=A40',
       'purpose=A41', 'purpose=A410', 'purpose=A42', 'purpose=A43',
       'purpose=A44', 'purpose=A45', 'purpose=A46', 'purpose=A48',
       'purpose=A49', 'savings=A61', 'savings=A62', 'savings=A63',
       'savings=A64', 'savings=A65', 'employment=A71', 'employment=A72',
       'employment=A73', 'employment=A74', 'employment=A75',
       'other_debtors=A101', 'other_debtors=A102', 'other_debtors=A103',
       'property=A121', 'property=A122', 'property=A123', 'property=A124',
       'installment_plans=A141', 'installment_plans=A142',
       'installment_plans=A143', 'housing=A151', 'housing=A152',
       'housing=A153', 'skill_level=A171', 'skill_level=A172',
       'skill_level=A173', 'skill_level=A174', 'telephone=A191',
       'telephone=A192', 'foreign_worker=A201', 'foreign_worker=A202',
       'credit'],
      dtype='object')
```

Out[6]:

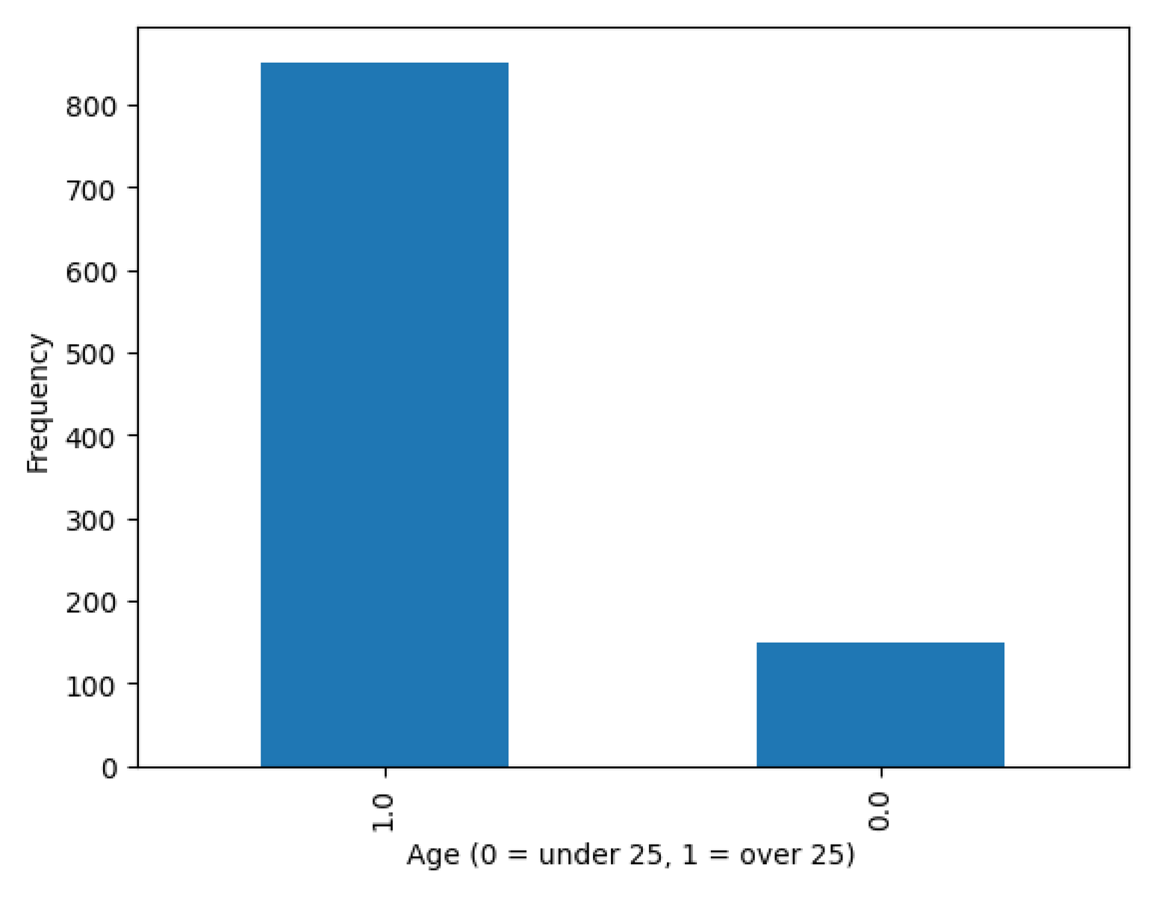
	month	credit_amount	investment_as_income_percentage	residence_since	age	number_of_credits	people_liable_for	status=A11	status=A12	status=A13	...	housing=A153	skill_level=A171	skill_level=A172	skill_level=A173	skill_level=A174	telephone=A
0	6.0	1169.0	4.0	4.0	1.0	2.0	1.0	1.0	0.0	0.0	...	0.0	0.0	0.0	1.0	0.0	
1	48.0	5951.0	2.0	2.0	0.0	1.0	1.0	0.0	1.0	0.0	...	0.0	0.0	0.0	1.0	0.0	
2	12.0	2096.0	2.0	3.0	1.0	1.0	2.0	0.0	0.0	0.0	...	0.0	0.0	1.0	0.0	0.0	
3	42.0	7882.0	2.0	4.0	1.0	1.0	2.0	1.0	0.0	0.0	...	1.0	0.0	0.0	1.0	0.0	
4	24.0	4870.0	3.0	4.0	1.0	2.0	2.0	1.0	0.0	0.0	...	1.0	0.0	0.0	1.0	0.0	

5 rows × 58 columns

In [7]:

```python
df['age'].value_counts().plot(kind='bar')
plt.xlabel("Age (0 = under 25, 1 = over 25)")
plt.ylabel("Frequency")
```

Out[7]: Text(0, 0.5, 'Frequency')

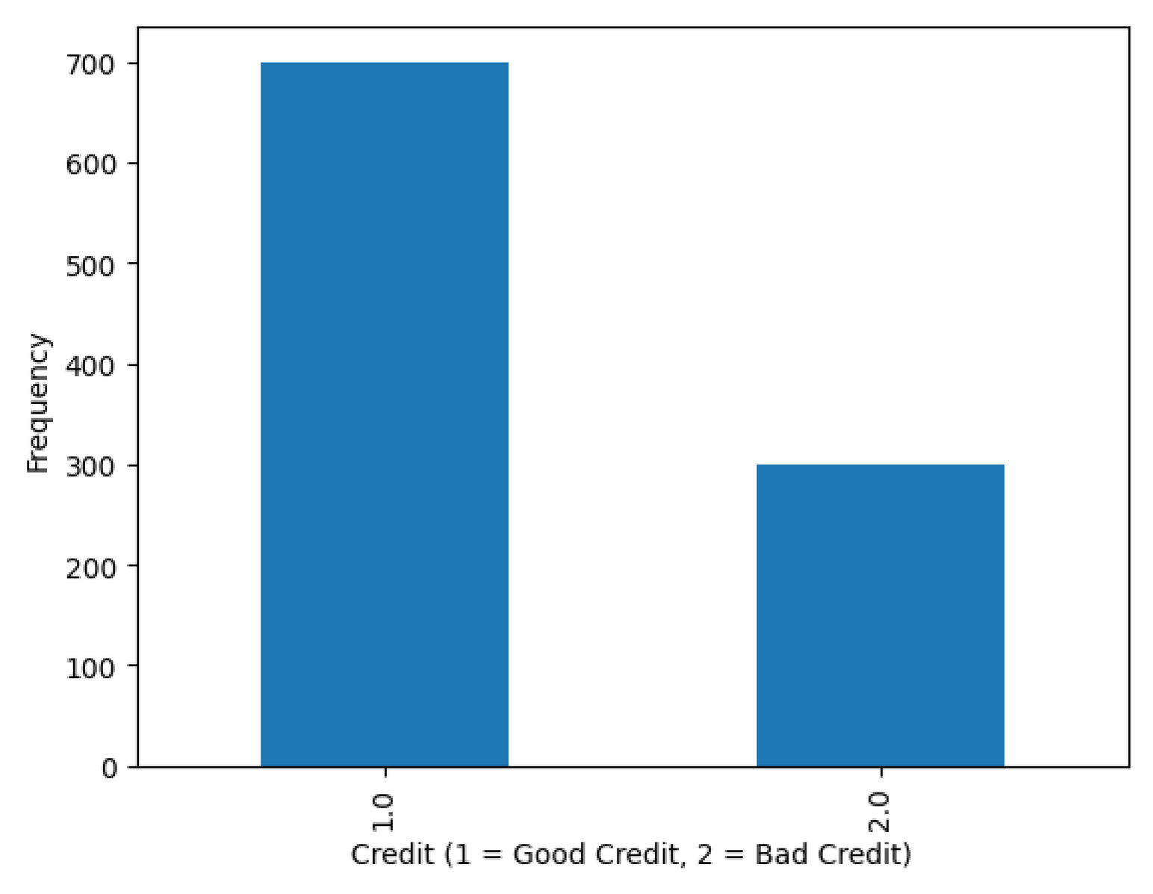


In [8]:

```python
print("Key: ", dataset_orig.metadata['label_maps'])
df['credit'].value_counts().plot(kind='bar')
plt.xlabel("Credit (1 = Good Credit, 2 = Bad Credit)")
plt.ylabel("Frequency")
```

```
Key:  [{1.0: 'Good Credit', 2.0: 'Bad Credit'}]
```

Out[8]: Text(0, 0.5, 'Frequency')



In [9]:

```python
metric_orig_train = BinaryLabelDatasetMetric(dataset_orig_train,
                                             unprivileged_groups=unprivileged_groups,
                                             privileged_groups=privileged_groups)
print("Original training dataset")
print("Difference in mean outcomes between unprivileged and privileged groups = %f" % metric_orig_train.mean_difference())
```

```
Original training dataset
Difference in mean outcomes between unprivileged and privileged groups = -0.169905
```

In [10]:

```python
print("Original training dataset")
print("Disparate Impact = %f" % metric_orig_train.disparate_impact())
```

```
Original training dataset
Disparate Impact = 0.766430
```

In [11]:

```python
print("Original training dataset")
print("Disparate Impact = %f" % metric_orig_train.disparate_impact())
```

```
Original training dataset
Disparate Impact = 0.766430
```

In [12]:

```python
text_expl = MetricTextExplainer(metric_orig_train)
json_expl = MetricJSONExplainer(metric_orig_train)
```

In [13]:

```python
print(text_expl.mean_difference())
```

```
Mean difference (mean label value on unprivileged instances - mean label value on privileged instances): -0.1699054740619017
```

In [14]:

```python
print(text_expl.disparate_impact())
```

```
Disparate impact (probability of favorable outcome for unprivileged instances / probability of favorable outcome for privileged instances): 0.7664297113013201
```

In [15]:

```python
def format_json(json_str):
    return json.dumps(json.loads(json_str, object_pairs_hook=OrderedDict), indent=2)
```

In [16]:

```python
print(format_json(json_expl.mean_difference()))
```


```
{
  "metric": "Mean Difference",
  "message": "Mean difference (mean label value on unprivileged instances - mean label value on privileged instances): -0.1699054740619017",
  "numPositivesUnprivileged": 63.0,
  "numInstancesUnprivileged": 113.0,
  "numPositivesPrivileged": 427.0,
  "numInstancesPrivileged": 587.0,
  "description": "Computed as the difference of the rate of favorable outcomes received by the unprivileged group to the privileged group.",
  "ideal": "The ideal value of this metric is 0.0"
}
```


In [17]:

```python
RW = Reweighing(unprivileged_groups=unprivileged_groups,
                privileged_groups=privileged_groups)
dataset_transf_train = RW.fit_transform(dataset_orig_train)
```

In [18]:

```python
dataset_transf_train.instance_weights
```

Out[18]:

```
array([0.96229508, 0.96229508, 0.96229508, 0.96229508, 0.96229508,
       0.96229508, 0.96229508, 0.96229508, 1.25555556, 0.678     ,
       1.100625  , 1.100625  , 0.96229508, 0.96229508, 1.100625  ,
       0.96229508, 1.25555556, 1.100625  , 0.96229508, 0.96229508,
       0.96229508, 0.96229508, 0.96229508, 0.96229508, 0.96229508,
       1.100625  , 0.96229508, 0.96229508, 0.96229508, 0.678     ,
       0.96229508, 0.96229508, 0.678     , 1.100625  , 0.96229508,
       0.678     , 0.96229508, 0.96229508, 1.100625  , 0.96229508,
       1.100625  , 0.96229508, 0.96229508, 1.25555556, 0.96229508,
       0.678     , 1.100625  , 0.96229508, 0.96229508, 1.25555556,
       1.100625  , 1.100625  , 0.96229508, 0.96229508, 1.100625  ,
       0.96229508, 0.96229508, 0.96229508, 0.96229508, 0.96229508,
       1.100625  , 0.96229508, 0.96229508, 0.96229508, 0.96229508,
       0.96229508, 0.96229508, 1.100625  , 0.678     , 0.96229508,
       0.96229508, 0.96229508, 0.96229508, 0.96229508, 1.25555556,
       1.100625  , 0.96229508, 1.100625  , 1.100625  , 0.96229508,
       0.96229508, 0.96229508, 1.25555556, 0.96229508, 0.96229508,
       0.96229508, 0.678     , 0.96229508, 0.96229508, 0.96229508,
       0.96229508, 0.96229508, 0.96229508, 0.96229508, 0.96229508,
       1.100625  , 0.96229508, 0.96229508, 1.25555556, 0.96229508,
       0.96229508, 1.25555556, 0.96229508, 0.96229508, 0.96229508,
       0.96229508, 0.96229508, 1.100625  , 1.100625  , 0.96229508,
       0.96229508, 1.100625  , 1.100625  , 1.25555556, 0.96229508,
       0.96229508, 1.100625  , 0.96229508, 0.96229508, 1.25555556,
       1.100625  , 0.96229508, 1.25555556, 1.100625  , 0.96229508,
       0.96229508, 1.25555556, 0.96229508, 0.96229508, 0.96229508,
       0.678     , 0.96229508, 0.678     , 0.96229508, 0.96229508,
       1.100625  , 0.96229508, 0.96229508, 0.96229508, 0.96229508,
       1.25555556, 0.96229508, 0.96229508, 0.678     , 1.100625  ,
       1.100625  , 0.96229508, 0.96229508, 1.100625  , 0.96229508,
       0.96229508, 0.96229508, 0.96229508, 1.100625  , 0.96229508,
       0.96229508, 1.100625  , 0.96229508, 0.96229508, 0.96229508,
       1.100625  , 0.96229508, 0.96229508, 1.100625  , 1.100625  ,
       1.100625  , 1.100625  , 0.96229508, 1.100625  , 0.96229508,
       0.96229508, 1.25555556, 0.96229508, 1.100625  , 0.96229508,
       0.678     , 1.100625  , 1.100625  , 0.96229508, 1.25555556,
       0.96229508, 0.96229508, 0.96229508, 1.100625  , 0.96229508,
       0.678     , 0.96229508, 0.96229508, 0.96229508, 0.96229508,
       1.100625  , 1.100625  , 1.100625  , 0.96229508, 0.96229508,
       0.96229508, 0.678     , 0.96229508, 1.100625  , 0.96229508,
       0.96229508, 1.100625  , 0.678     , 1.100625  , 1.25555556,
       0.96229508, 1.100625  , 0.96229508, 0.96229508, 0.96229508,
       1.100625  , 0.96229508, 1.100625  , 0.96229508, 0.96229508,
       1.25555556, 0.96229508, 0.96229508, 0.96229508, 0.678     ,
       0.678     , 0.96229508, 1.25555556, 0.96229508, 0.96229508,
       0.96229508, 1.25555556, 1.100625  , 1.100625  , 1.25555556,
       0.96229508, 0.678     , 1.100625  , 0.96229508, 1.100625  ,
       1.25555556, 0.96229508, 0.96229508, 0.96229508, 1.100625  ,
       1.100625  , 0.96229508, 0.96229508, 1.100625  , 0.96229508,
       0.96229508, 0.96229508, 0.96229508, 0.96229508, 0.96229508,
       0.96229508, 0.96229508, 0.96229508, 0.96229508, 0.96229508,
       0.96229508, 0.96229508, 0.96229508, 0.96229508, 0.96229508,
       0.96229508, 0.96229508, 1.25555556, 0.96229508, 0.96229508,
       1.100625  , 0.96229508, 0.678     , 0.96229508, 1.100625  ,
       1.100625  , 1.100625  , 0.96229508, 0.96229508, 0.96229508,
       0.96229508, 0.96229508, 0.96229508, 1.100625  , 1.100625  ,
       0.96229508, 0.96229508, 0.96229508, 0.96229508, 1.100625  ,
       1.25555556, 0.96229508, 0.96229508, 0.96229508, 0.96229508,
       0.96229508, 0.96229508, 1.25555556, 0.96229508, 1.100625  ,
       1.25555556, 0.678     , 0.96229508, 1.100625  , 1.100625  ,
       1.100625  , 1.100625  , 0.96229508, 0.96229508, 0.96229508,
       1.25555556, 1.100625  , 1.100625  , 1.100625  , 0.96229508,
       1.100625  , 1.100625  , 0.96229508, 0.96229508, 1.25555556,
       0.96229508, 1.25555556, 0.96229508, 0.96229508, 0.96229508,
       0.96229508, 0.96229508, 0.96229508, 1.100625  , 0.96229508,
       1.100625  , 0.96229508, 1.100625  , 0.96229508, 0.96229508,
       0.96229508, 0.96229508, 0.96229508, 0.96229508, 0.96229508,
       0.96229508, 1.100625  , 0.96229508, 1.100625  , 1.25555556,
       1.100625  , 0.96229508, 0.678     , 1.25555556, 0.96229508,
       0.678     , 1.100625  , 0.96229508, 1.100625  , 1.100625  ,
       0.96229508, 0.96229508, 0.96229508, 0.96229508, 0.96229508,
       0.96229508, 0.96229508, 1.100625  , 1.100625  , 1.25555556,
       1.25555556, 0.96229508, 1.100625  , 1.100625  , 0.96229508,
       0.96229508, 1.100625  , 0.96229508, 1.100625  , 0.678     ,
       0.96229508, 1.100625  , 0.96229508, 1.25555556, 0.96229508,
       1.100625  , 0.678     , 0.96229508, 0.96229508, 0.96229508,
       0.678     , 1.25555556, 0.96229508, 1.25555556, 0.678     ,
       1.100625  , 1.100625  , 0.96229508, 0.96229508, 0.678     ,
       0.96229508, 0.96229508, 0.96229508, 0.96229508, 1.100625  ,
       0.96229508, 0.96229508, 0.678     , 1.25555556, 1.100625  ,
       0.96229508, 0.96229508, 0.678     , 0.96229508, 0.96229508,
       1.100625  , 1.100625  , 0.96229508, 1.100625  , 1.100625  ,
       0.96229508, 0.96229508, 0.96229508, 0.96229508, 0.96229508,
       0.96229508, 0.96229508, 0.96229508, 0.96229508, 0.678     ,
       0.96229508, 0.96229508, 0.96229508, 1.25555556, 0.96229508,
       0.96229508, 0.96229508, 0.678     , 0.96229508, 1.100625  ,
       0.96229508, 0.96229508, 0.96229508, 0.96229508, 0.96229508,
       0.96229508, 1.100625  , 0.96229508, 1.100625  , 0.678     ,
       0.96229508, 0.96229508, 0.96229508, 0.96229508, 0.96229508,
       0.96229508, 0.96229508, 1.25555556, 0.678     , 1.100625  ,
       0.96229508, 1.100625  , 0.96229508, 1.100625  , 0.96229508,
       1.25555556, 0.96229508, 0.96229508, 1.25555556, 1.100625  ,
       1.100625  , 1.100625  , 0.96229508, 0.96229508, 0.96229508,
       0.96229508, 1.25555556, 0.96229508, 0.96229508, 0.678     ,
       1.100625  , 0.96229508, 0.96229508, 0.96229508, 0.96229508,
       0.96229508, 1.25555556, 1.100625  , 0.96229508, 0.96229508,
       0.96229508, 1.25555556, 0.96229508, 0.96229508, 1.100625  ,
       0.96229508, 1.100625  , 1.25555556, 0.96229508, 1.100625  ,
       1.25555556, 0.96229508, 1.100625  , 0.96229508, 0.96229508,
       1.100625  , 1.100625  , 0.96229508, 0.96229508, 1.100625  ,
       1.100625  , 0.96229508, 0.96229508, 0.96229508, 0.678     ,
       0.96229508, 0.96229508, 0.96229508, 1.100625  , 0.96229508,
       0.96229508, 0.96229508, 0.96229508, 1.25555556, 0.96229508,
       0.96229508, 0.96229508, 0.96229508, 0.96229508, 0.96229508,
       0.96229508, 1.100625  , 1.100625  , 0.96229508, 0.96229508,
       0.96229508, 1.25555556, 0.96229508, 0.96229508, 0.96229508,
       0.96229508, 1.100625  , 0.96229508, 0.96229508, 1.25555556,
       0.96229508, 1.100625  , 1.100625  , 0.96229508, 0.96229508,
       1.25555556, 0.96229508, 0.96229508, 0.96229508, 0.96229508,
       0.678     , 0.96229508, 1.100625  , 1.25555556, 0.96229508,
       0.96229508, 0.96229508, 1.25555556, 0.678     , 0.96229508,
       0.96229508, 1.25555556, 0.678     , 0.96229508, 0.96229508,
       0.96229508, 0.96229508, 0.96229508, 0.96229508, 1.100625  ,
       0.96229508, 0.678     , 1.100625  , 1.100625  , 1.100625  ,
       0.96229508, 1.100625  , 0.96229508, 1.100625  , 0.96229508,
       0.96229508, 0.96229508, 0.96229508, 0.678     , 1.100625  ,
       0.96229508, 0.96229508, 0.678     , 1.25555556, 0.96229508,
       0.96229508, 0.96229508, 1.100625  , 0.96229508, 1.100625  ,
       1.100625  , 0.96229508, 0.678     , 0.96229508, 0.96229508,
       1.100625  , 0.96229508, 0.96229508, 0.96229508, 0.96229508,
       1.100625  , 0.96229508, 0.678     , 1.100625  , 1.100625  ,
       0.678     , 0.96229508, 1.100625  , 0.96229508, 0.678     ,
       0.96229508, 1.25555556, 0.678     , 1.25555556, 0.96229508,
       0.96229508, 0.96229508, 1.100625  , 0.96229508, 0.96229508,
       1.100625  , 1.100625  , 0.96229508, 0.96229508, 0.96229508,
       0.96229508, 0.96229508, 0.96229508, 0.96229508, 1.100625  ,
       0.96229508, 0.96229508, 0.96229508, 0.96229508, 0.96229508,
       0.678     , 1.25555556, 0.96229508, 0.96229508, 0.96229508,
       0.96229508, 0.96229508, 0.96229508, 0.678     , 0.96229508,
       0.96229508, 1.100625  , 0.678     , 1.100625  , 1.100625  ,
       1.25555556, 0.96229508, 0.96229508, 0.96229508, 0.96229508,
       0.96229508, 1.100625  , 0.96229508, 0.96229508, 1.100625  ,
       1.100625  , 0.96229508, 0.96229508, 0.96229508, 0.96229508,
       1.100625  , 0.96229508, 1.25555556, 1.100625  , 0.96229508,
       0.678     , 0.96229508, 1.25555556, 0.96229508, 0.96229508,
       0.678     , 1.25555556, 1.25555556, 0.96229508, 0.96229508,
       0.96229508, 1.100625  , 0.96229508, 1.100625  , 0.96229508,
       1.100625  , 1.100625  , 1.100625  , 1.100625  , 0.96229508,
       0.96229508, 1.100625  , 1.100625  , 0.96229508, 0.96229508,
       0.96229508, 1.25555556, 1.25555556, 0.96229508, 0.96229508])
```

In [19]:

```python
len(dataset_transf_train.instance_weights)
```

Out[19]: 700

In [21]:

```python
metric_transf_train = BinaryLabelDatasetMetric(dataset_transf_train,
                                               unprivileged_groups=unprivileged_groups,
                                               privileged_groups=privileged_groups)
print("Transformed training dataset")
print("Difference in mean outcomes between unprivileged and privileged groups = %f" % metric_transf_train.mean_difference())
```

```
Transformed training dataset
Difference in mean outcomes between unprivileged and privileged groups = 0.000000
```

In [22]:

```python
print("Transformed training dataset")
print("Disparate Impact = %f" % metric_transf_train.disparate_impact())
```

```
Transformed training dataset
Disparate Impact = 1.000000
```

In [ ]: