adversarial_debiasing_PG2374 (1)

December 23, 2022

0.1 Step 1 install package using github repository: https://github.com/Trusted-AI/AIF360.git

[1]:
```
!pip install git+https://github.com/pg2374/AIF.git
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Collecting git+https://github.com/pg2374/AIF.git
  Cloning https://github.com/pg2374/AIF.git to /tmp/pip-req-build-8ufx28dy
  Running command git clone -q https://github.com/pg2374/AIF.git /tmp/pip-req-
build-8ufx28dy
Requirement already satisfied: numpy>=1.16 in /usr/local/lib/python3.8/dist-
packages (from aif360==0.4.0) (1.21.6)
Collecting scipy<1.6.0,>=1.2.0
  Downloading scipy-1.5.4-cp38-cp38-manylinux1_x86_64.whl (25.8 MB)
     |                 | 25.8 MB 1.9 MB/s
Requirement already satisfied: pandas>=0.24.0 in
/usr/local/lib/python3.8/dist-packages (from aif360==0.4.0) (1.3.5)
Requirement already satisfied: scikit-learn>=0.22.1 in
/usr/local/lib/python3.8/dist-packages (from aif360==0.4.0) (1.0.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.8/dist-
packages (from aif360==0.4.0) (3.2.2)
Collecting tempeh
  Downloading tempeh-0.1.12-py3-none-any.whl (39 kB)
Requirement already satisfied: python-dateutil>=2.7.3 in
/usr/local/lib/python3.8/dist-packages (from pandas>=0.24.0->aif360==0.4.0)
(2.8.2)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.8/dist-
packages (from pandas>=0.24.0->aif360==0.4.0) (2022.6)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.8/dist-
packages (from python-dateutil>=2.7.3->pandas>=0.24.0->aif360==0.4.0) (1.15.0)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.8/dist-
packages (from scikit-learn>=0.22.1->aif360==0.4.0) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.8/dist-packages (from scikit-
learn>=0.22.1->aif360==0.4.0) (3.1.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.8/dist-packages (from matplotlib->aif360==0.4.0) (1.4.4)
```

```
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
/usr/local/lib/python3.8/dist-packages (from matplotlib->aif360==0.4.0) (3.0.9)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.8/dist-
packages (from matplotlib->aif360==0.4.0) (0.11.0)
Collecting shap
  Downloading
shap-0.41.0-cp38-cp38-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (575 kB)
     |                 | 575 kB 58.7 MB/s
Requirement already satisfied: requests in /usr/local/lib/python3.8/dist-
packages (from tempeh->aif360==0.4.0) (2.23.0)
Requirement already satisfied: pytest in /usr/local/lib/python3.8/dist-packages
(from tempeh->aif360==0.4.0) (3.6.4)
Collecting memory-profiler
  Downloading memory_profiler-0.61.0-py3-none-any.whl (31 kB)
Requirement already satisfied: psutil in /usr/local/lib/python3.8/dist-packages
(from memory-profiler->tempeh->aif360==0.4.0) (5.4.8)
Requirement already satisfied: atomicwrites>=1.0 in
/usr/local/lib/python3.8/dist-packages (from pytest->tempeh->aif360==0.4.0)
(1.4.1)
Requirement already satisfied: setuptools in /usr/local/lib/python3.8/dist-
packages (from pytest->tempeh->aif360==0.4.0) (57.4.0)
Requirement already satisfied: pluggy<0.8,>=0.5 in
/usr/local/lib/python3.8/dist-packages (from pytest->tempeh->aif360==0.4.0)
(0.7.1)
Requirement already satisfied: py>=1.5.0 in /usr/local/lib/python3.8/dist-
packages (from pytest->tempeh->aif360==0.4.0) (1.11.0)
Requirement already satisfied: more-itertools>=4.0.0 in
/usr/local/lib/python3.8/dist-packages (from pytest->tempeh->aif360==0.4.0)
(9.0.0)
Requirement already satisfied: attrs>=17.4.0 in /usr/local/lib/python3.8/dist-
packages (from pytest->tempeh->aif360==0.4.0) (22.1.0)
Requirement already satisfied: chardet<4,>=3.0.2 in
/usr/local/lib/python3.8/dist-packages (from requests->tempeh->aif360==0.4.0)
(3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.8/dist-packages (from requests->tempeh->aif360==0.4.0)
(2022.12.7)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.8/dist-
packages (from requests->tempeh->aif360==0.4.0) (2.10)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/usr/local/lib/python3.8/dist-packages (from requests->tempeh->aif360==0.4.0)
(1.24.3)
Requirement already satisfied: tqdm>4.25.0 in /usr/local/lib/python3.8/dist-
packages (from shap->tempeh->aif360==0.4.0) (4.64.1)
Requirement already satisfied: packaging>20.9 in /usr/local/lib/python3.8/dist-
packages (from shap->tempeh->aif360==0.4.0) (21.3)
Collecting slicer==0.0.7
  Downloading slicer-0.0.7-py3-none-any.whl (14 kB)
```

```
Requirement already satisfied: numba in /usr/local/lib/python3.8/dist-packages
(from shap->tempeh->aif360==0.4.0) (0.56.4)
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.8/dist-
packages (from shap->tempeh->aif360==0.4.0) (1.5.0)
Requirement already satisfied: llvmlite<0.40,>=0.39.0dev0 in
/usr/local/lib/python3.8/dist-packages (from numba->shap->tempeh->aif360==0.4.0)
(0.39.1)
Requirement already satisfied: importlib-metadata in
/usr/local/lib/python3.8/dist-packages (from numba->shap->tempeh->aif360==0.4.0)
(5.1.0)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.8/dist-
packages (from importlib-metadata->numba->shap->tempeh->aif360==0.4.0) (3.11.0)
Building wheels for collected packages: aif360
  Building wheel for aif360 (setup.py) … done
  Created wheel for aif360: filename=aif360-0.4.0-py3-none-any.whl size=599953
sha256=f35861d5cbad7c0899fe32336bf34e76f199101f67cbf6bc3a7ce636cd177abf
  Stored in directory: /tmp/pip-ephem-wheel-cache-
qz_rseve/wheels/71/13/55/3e9ee7172da0c15a9c49057274435b60135ce5818112294d4b
Successfully built aif360
Installing collected packages: scipy, slicer, shap, memory-profiler, tempeh,
aif360
  Attempting uninstall: scipy
    Found existing installation: scipy 1.7.3
    Uninstalling scipy-1.7.3:
      Successfully uninstalled scipy-1.7.3
ERROR: pip's dependency resolver does not currently take into account all

the packages that are installed. This behaviour is the source of the following

dependency conflicts.

xarray-einstats 0.4.0 requires scipy>=1.6, but you have scipy 1.5.4 which is

incompatible.
Successfully installed aif360-0.4.0 memory-profiler-0.61.0 scipy-1.5.4
shap-0.41.0 slicer-0.0.7 tempeh-0.1.12
```

[2]:
```
!pip install fairlearn
!pip install tensorflow
!pip install scikit-plot
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Collecting fairlearn
  Downloading fairlearn-0.8.0-py3-none-any.whl (235 kB)
     |                  | 235 kB 14.3 MB/s
Requirement already satisfied: numpy>=1.17.2 in
/usr/local/lib/python3.8/dist-packages (from fairlearn) (1.21.6)
Requirement already satisfied: scikit-learn>=0.22.1 in
/usr/local/lib/python3.8/dist-packages (from fairlearn) (1.0.2)
```

```
Requirement already satisfied: pandas>=0.25.1 in /usr/local/lib/python3.8/dist-
packages (from fairlearn) (1.3.5)
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.8/dist-
packages (from fairlearn) (1.5.4)
Requirement already satisfied: python-dateutil>=2.7.3 in
/usr/local/lib/python3.8/dist-packages (from pandas>=0.25.1->fairlearn) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.8/dist-
packages (from pandas>=0.25.1->fairlearn) (2022.6)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.8/dist-
packages (from python-dateutil>=2.7.3->pandas>=0.25.1->fairlearn) (1.15.0)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.8/dist-
packages (from scikit-learn>=0.22.1->fairlearn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.8/dist-packages (from scikit-learn>=0.22.1->fairlearn)
(3.1.0)
Installing collected packages: fairlearn
Successfully installed fairlearn-0.8.0
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: tensorflow in /usr/local/lib/python3.8/dist-
packages (2.9.2)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in
/usr/local/lib/python3.8/dist-packages (from tensorflow) (0.4.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
/usr/local/lib/python3.8/dist-packages (from tensorflow) (1.51.1)
Requirement already satisfied: termcolor>=1.1.0 in
/usr/local/lib/python3.8/dist-packages (from tensorflow) (2.1.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
/usr/local/lib/python3.8/dist-packages (from tensorflow) (0.28.0)
Requirement already satisfied: opt-einsum>=2.3.2 in
/usr/local/lib/python3.8/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied: google-pasta>=0.1.1 in
/usr/local/lib/python3.8/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.8/dist-
packages (from tensorflow) (1.21.6)
Requirement already satisfied: protobuf<3.20,>=3.9.2 in
/usr/local/lib/python3.8/dist-packages (from tensorflow) (3.19.6)
Requirement already satisfied: libclang>=13.0.0 in
/usr/local/lib/python3.8/dist-packages (from tensorflow) (14.0.6)
Requirement already satisfied: astunparse>=1.6.0 in
/usr/local/lib/python3.8/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: tensorflow-estimator<2.10.0,>=2.9.0rc0 in
/usr/local/lib/python3.8/dist-packages (from tensorflow) (2.9.0)
Requirement already satisfied: tensorboard<2.10,>=2.9 in
/usr/local/lib/python3.8/dist-packages (from tensorflow) (2.9.1)
Requirement already satisfied: flatbuffers<2,>=1.12 in
/usr/local/lib/python3.8/dist-packages (from tensorflow) (1.12)
Requirement already satisfied: packaging in /usr/local/lib/python3.8/dist-
```

```
packages (from tensorflow) (21.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.8/dist-
packages (from tensorflow) (57.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in
/usr/local/lib/python3.8/dist-packages (from tensorflow) (4.4.0)
Requirement already satisfied: keras<2.10.0,>=2.9.0rc0 in
/usr/local/lib/python3.8/dist-packages (from tensorflow) (2.9.0)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.8/dist-
packages (from tensorflow) (3.1.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.8/dist-
packages (from tensorflow) (1.3.0)
Requirement already satisfied: keras-preprocessing>=1.1.1 in
/usr/local/lib/python3.8/dist-packages (from tensorflow) (1.1.2)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.8/dist-
packages (from tensorflow) (1.14.1)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.8/dist-
packages (from tensorflow) (1.15.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
/usr/local/lib/python3.8/dist-packages (from astunparse>=1.6.0->tensorflow)
(0.38.4)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in
/usr/local/lib/python3.8/dist-packages (from tensorboard<2.10,>=2.9->tensorflow)
(0.4.6)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in
/usr/local/lib/python3.8/dist-packages (from tensorboard<2.10,>=2.9->tensorflow)
(0.6.1)
Requirement already satisfied: requests<3,>=2.21.0 in
/usr/local/lib/python3.8/dist-packages (from tensorboard<2.10,>=2.9->tensorflow)
(2.23.0)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in
/usr/local/lib/python3.8/dist-packages (from tensorboard<2.10,>=2.9->tensorflow)
(1.8.1)
Requirement already satisfied: google-auth<3,>=1.6.3 in
/usr/local/lib/python3.8/dist-packages (from tensorboard<2.10,>=2.9->tensorflow)
(2.15.0)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.8/dist-
packages (from tensorboard<2.10,>=2.9->tensorflow) (3.4.1)
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.8/dist-
packages (from tensorboard<2.10,>=2.9->tensorflow) (1.0.1)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in
/usr/local/lib/python3.8/dist-packages (from google-
auth<3,>=1.6.3->tensorboard<2.10,>=2.9->tensorflow) (5.2.0)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/usr/local/lib/python3.8/dist-packages (from google-
auth<3,>=1.6.3->tensorboard<2.10,>=2.9->tensorflow) (0.2.8)
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.8/dist-
packages (from google-auth<3,>=1.6.3->tensorboard<2.10,>=2.9->tensorflow) (4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
```

```
/usr/local/lib/python3.8/dist-packages (from google-auth-
oauthlib<0.5,>=0.4.1->tensorboard<2.10,>=2.9->tensorflow) (1.3.1)
Requirement already satisfied: importlib-metadata>=4.4 in
/usr/local/lib/python3.8/dist-packages (from
markdown>=2.6.8->tensorboard<2.10,>=2.9->tensorflow) (5.1.0)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.8/dist-
packages (from importlib-
metadata>=4.4->markdown>=2.6.8->tensorboard<2.10,>=2.9->tensorflow) (3.11.0)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in
/usr/local/lib/python3.8/dist-packages (from pyasn1-modules>=0.2.1->google-
auth<3,>=1.6.3->tensorboard<2.10,>=2.9->tensorflow) (0.4.8)
Requirement already satisfied: chardet<4,>=3.0.2 in
/usr/local/lib/python3.8/dist-packages (from
requests<3,>=2.21.0->tensorboard<2.10,>=2.9->tensorflow) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.8/dist-packages (from
requests<3,>=2.21.0->tensorboard<2.10,>=2.9->tensorflow) (2022.12.7)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.8/dist-
packages (from requests<3,>=2.21.0->tensorboard<2.10,>=2.9->tensorflow) (2.10)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/usr/local/lib/python3.8/dist-packages (from
requests<3,>=2.21.0->tensorboard<2.10,>=2.9->tensorflow) (1.24.3)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.8/dist-
packages (from requests-oauthlib>=0.7.0->google-auth-
oauthlib<0.5,>=0.4.1->tensorboard<2.10,>=2.9->tensorflow) (3.2.2)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in
/usr/local/lib/python3.8/dist-packages (from packaging->tensorflow) (3.0.9)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Collecting scikit-plot
  Downloading scikit_plot-0.3.7-py3-none-any.whl (33 kB)
Requirement already satisfied: scikit-learn>=0.18 in
/usr/local/lib/python3.8/dist-packages (from scikit-plot) (1.0.2)
Requirement already satisfied: scipy>=0.9 in /usr/local/lib/python3.8/dist-
packages (from scikit-plot) (1.5.4)
Requirement already satisfied: joblib>=0.10 in /usr/local/lib/python3.8/dist-
packages (from scikit-plot) (1.2.0)
Requirement already satisfied: matplotlib>=1.4.0 in
/usr/local/lib/python3.8/dist-packages (from scikit-plot) (3.2.2)
Requirement already satisfied: python-dateutil>=2.1 in
/usr/local/lib/python3.8/dist-packages (from matplotlib>=1.4.0->scikit-plot)
(2.8.2)
Requirement already satisfied: numpy>=1.11 in /usr/local/lib/python3.8/dist-
packages (from matplotlib>=1.4.0->scikit-plot) (1.21.6)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.8/dist-
packages (from matplotlib>=1.4.0->scikit-plot) (0.11.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
/usr/local/lib/python3.8/dist-packages (from matplotlib>=1.4.0->scikit-plot)
```

```
(3.0.9)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.8/dist-packages (from matplotlib>=1.4.0->scikit-plot)
(1.4.4)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.8/dist-
packages (from python-dateutil>=2.1->matplotlib>=1.4.0->scikit-plot) (1.15.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.8/dist-packages (from scikit-learn>=0.18->scikit-plot)
(3.1.0)
Installing collected packages: scikit-plot
Successfully installed scikit-plot-0.3.7
```

1 Step 2: Download the data files

[3]:
```
!wget https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data
!wget https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.test
!wget https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.
 ↪names
```

```
--2022-12-23 06:04:05--  https://archive.ics.uci.edu/ml/machine-learning-
databases/adult/adult.data
Resolving archive.ics.uci.edu (archive.ics.uci.edu)… 128.195.10.252
Connecting to archive.ics.uci.edu (archive.ics.uci.edu)|128.195.10.252|:443…
connected.
HTTP request sent, awaiting response… 200 OK
Length: 3974305 (3.8M) [application/x-httpd-php]
Saving to: ‘adult.data’

adult.data          100%[===================>]   3.79M  3.07MB/s    in 1.2s

2022-12-23 06:04:08 (3.07 MB/s) - ‘adult.data’ saved [3974305/3974305]

--2022-12-23 06:04:08--  https://archive.ics.uci.edu/ml/machine-learning-
databases/adult/adult.test
Resolving archive.ics.uci.edu (archive.ics.uci.edu)… 128.195.10.252
Connecting to archive.ics.uci.edu (archive.ics.uci.edu)|128.195.10.252|:443…
connected.
HTTP request sent, awaiting response… 200 OK
Length: 2003153 (1.9M) [application/x-httpd-php]
Saving to: ‘adult.test’

adult.test          100%[===================>]   1.91M  1.77MB/s    in 1.1s

2022-12-23 06:04:10 (1.77 MB/s) - ‘adult.test’ saved [2003153/2003153]

--2022-12-23 06:04:10--  https://archive.ics.uci.edu/ml/machine-learning-
databases/adult/adult.names
```

```
Resolving archive.ics.uci.edu (archive.ics.uci.edu)… 128.195.10.252
Connecting to archive.ics.uci.edu (archive.ics.uci.edu)|128.195.10.252|:443…
connected.
HTTP request sent, awaiting response… 200 OK
Length: 5229 (5.1K) [application/x-httpd-php]
Saving to: ‘adult.names’

adult.names         100%[===================>]   5.11K  --.-KB/s    in 0s

2022-12-23 06:04:11 (114 MB/s) - ‘adult.names’ saved [5229/5229]
```

2 Step 3: Copy the data files to the location specified in the package

[4]:
```
!cp adult.data /usr/local/lib/python3.8/dist-packages/aif360/data/raw/adult/
!cp adult.test /usr/local/lib/python3.8/dist-packages/aif360/data/raw/adult/
!cp adult.names /usr/local/lib/python3.8/dist-packages/aif360/data/raw/adult/
```

This notebook demonstrates the use of adversarial debiasing algorithm to learn a fair classifier. Adversarial debiasing [1] is an in-processing technique that learns a classifier to maximize prediction accuracy and simultaneously reduce an adversary's ability to determine the protected attribute from the predictions. This approach leads to a fair classifier as the predictions cannot carry any group discrimination information that the adversary can exploit. We will see how to use this algorithm for learning models with and without fairness constraints and apply them on the Adult dataset.

[5]:
```
%matplotlib inline
# Load all necessary packages
import sys
sys.path.append("../")
from aif360.datasets import BinaryLabelDataset
from aif360.datasets import AdultDataset, GermanDataset, CompasDataset
from aif360.metrics import BinaryLabelDatasetMetric
from aif360.metrics import ClassificationMetric
from aif360.metrics.utils import compute_boolean_conditioning_vector

from aif360.algorithms.preprocessing.optim_preproc_helpers.
 ↪data_preproc_functions import load_preproc_data_adult,␣
 ↪load_preproc_data_compas, load_preproc_data_german

from aif360.algorithms.inprocessing.adversarial_debiasing import␣
 ↪AdversarialDebiasing

from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler, MaxAbsScaler, MinMaxScaler
from sklearn.metrics import accuracy_score
from sklearn import metrics

from IPython.display import Markdown, display
import matplotlib.pyplot as plt

import tensorflow.compat.v1 as tf
tf.disable_eager_execution()

import scikitplot as skplt
```

Load dataset and set options

[6]:
```
# Get the dataset and split into train and test
dataset_orig = load_preproc_data_adult()

privileged_groups = [{'sex': 1}]
unprivileged_groups = [{'sex': 0}]

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

[7]:
```
# print out some labels, names, etc.
display(Markdown("#### Training Dataset shape"))
print(dataset_orig_train.features.shape)
display(Markdown("#### Test Dataset shape"))
print(dataset_orig_test.features.shape)
display(Markdown("#### Favorable and unfavorable labels"))
print(dataset_orig_train.favorable_label, dataset_orig_train.unfavorable_label)
display(Markdown("#### Protected attribute names"))
print(dataset_orig_train.protected_attribute_names)
display(Markdown("#### Privileged and unprivileged protected attribute values"))
print(dataset_orig_train.privileged_protected_attributes,
      dataset_orig_train.unprivileged_protected_attributes)
display(Markdown("#### Dataset feature names"))
print(dataset_orig_train.feature_names)
```

Training Dataset shape

```
(34189, 18)
```

Test Dataset shape

```
(14653, 18)
```

Favorable and unfavorable labels

```
1.0 0.0
```

Protected attribute names

```
['sex', 'race']
```

Privileged and unprivileged protected attribute values

```
[array([1.]), array([1.])] [array([0.]), array([0.])]
```

Dataset feature names

```
['race', 'sex', 'Age (decade)=10', 'Age (decade)=20', 'Age (decade)=30', 'Age
(decade)=40', 'Age (decade)=50', 'Age (decade)=60', 'Age (decade)=>=70',
'Education Years=6', 'Education Years=7', 'Education Years=8', 'Education
Years=9', 'Education Years=10', 'Education Years=11', 'Education Years=12',
'Education Years=<6', 'Education Years=>12']
```

[7]:

```
```

Metric for original training data

[8]:

```
# Metric for the original dataset
metric_orig_train = BinaryLabelDatasetMetric(dataset_orig_train,
                                                                          ␣
 ↪unprivileged_groups=unprivileged_groups,
                                                                          ␣
 ↪privileged_groups=privileged_groups)
display(Markdown("#### Original training dataset"))
print("Train set: Difference in mean outcomes between unprivileged and␣
 ↪privileged groups = %f" % metric_orig_train.mean_difference())
metric_orig_test = BinaryLabelDatasetMetric(dataset_orig_test,
                                                                          ␣
 ↪unprivileged_groups=unprivileged_groups,
                                                                          ␣
 ↪privileged_groups=privileged_groups)
print("Test set: Difference in mean outcomes between unprivileged and␣
 ↪privileged groups = %f" % metric_orig_test.mean_difference())
```

Original training dataset

```
Train set: Difference in mean outcomes between unprivileged and privileged
groups = -0.191948
Test set: Difference in mean outcomes between unprivileged and privileged groups
= -0.200418
```

3 Part 1: Original work by author

##In each part, we have added addtional metrics like precision and recall to gauge performance on imbalanced data ### Preprocessing dataset using MaxAbsScaler

```
[9]: max_abs_scaler = MaxAbsScaler()
dataset_orig_train.features = max_abs_scaler.fit_transform(dataset_orig_train.
  ↪features)
dataset_orig_test.features = max_abs_scaler.transform(dataset_orig_test.
  ↪features)
metric_scaled_train = BinaryLabelDatasetMetric(dataset_orig_train,
                             unprivileged_groups=unprivileged_groups,
                             privileged_groups=privileged_groups)
display(Markdown("#### Scaled dataset - Verify that the scaling does not affect␣
  ↪the group label statistics"))
print("Train set: Difference in mean outcomes between unprivileged and␣
  ↪privileged groups = %f" % metric_scaled_train.mean_difference())
metric_scaled_test = BinaryLabelDatasetMetric(dataset_orig_test,
                             unprivileged_groups=unprivileged_groups,
                             privileged_groups=privileged_groups)
print("Test set: Difference in mean outcomes between unprivileged and␣
  ↪privileged groups = %f" % metric_scaled_test.mean_difference())
```

Scaled dataset - Verify that the scaling does not affect the group label statistics

```
Train set: Difference in mean outcomes between unprivileged and privileged
groups = -0.191948
Test set: Difference in mean outcomes between unprivileged and privileged groups
= -0.200418
```

3.0.1 Learn plain classifier without debiasing

```
[10]: # Load post-processing algorithm that equalizes the odds
# Learn parameters with debias set to False
sess = tf.Session()
plain_model = AdversarialDebiasing(privileged_groups = privileged_groups,
                          unprivileged_groups = unprivileged_groups,
                          scope_name='plain_classifier_part_1',
                          debias=False,
                          sess=sess)
```

```
[11]: plain_model.fit(dataset_orig_train)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.8/dist-
packages/tensorflow/python/util/dispatch.py:1082: calling dropout (from
tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will be removed
in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 -
keep_prob`.
```

```
epoch 0; iter: 0; batch classifier loss: 0.720402
epoch 0; iter: 200; batch classifier loss: 0.407042
epoch 1; iter: 0; batch classifier loss: 0.361959
epoch 1; iter: 200; batch classifier loss: 0.437485
epoch 2; iter: 0; batch classifier loss: 0.480875
epoch 2; iter: 200; batch classifier loss: 0.456051
epoch 3; iter: 0; batch classifier loss: 0.405031
epoch 3; iter: 200; batch classifier loss: 0.341224
epoch 4; iter: 0; batch classifier loss: 0.335127
epoch 4; iter: 200; batch classifier loss: 0.392714
epoch 5; iter: 0; batch classifier loss: 0.491046
epoch 5; iter: 200; batch classifier loss: 0.359802
epoch 6; iter: 0; batch classifier loss: 0.407765
epoch 6; iter: 200; batch classifier loss: 0.339861
epoch 7; iter: 0; batch classifier loss: 0.412609
epoch 7; iter: 200; batch classifier loss: 0.466955
epoch 8; iter: 0; batch classifier loss: 0.375506
epoch 8; iter: 200; batch classifier loss: 0.368174
epoch 9; iter: 0; batch classifier loss: 0.452794
epoch 9; iter: 200; batch classifier loss: 0.360016
epoch 10; iter: 0; batch classifier loss: 0.357388
epoch 10; iter: 200; batch classifier loss: 0.372595
epoch 11; iter: 0; batch classifier loss: 0.407882
epoch 11; iter: 200; batch classifier loss: 0.429657
epoch 12; iter: 0; batch classifier loss: 0.439970
epoch 12; iter: 200; batch classifier loss: 0.397106
epoch 13; iter: 0; batch classifier loss: 0.414627
epoch 13; iter: 200; batch classifier loss: 0.494460
epoch 14; iter: 0; batch classifier loss: 0.447730
epoch 14; iter: 200; batch classifier loss: 0.301898
epoch 15; iter: 0; batch classifier loss: 0.382302
epoch 15; iter: 200; batch classifier loss: 0.453237
epoch 16; iter: 0; batch classifier loss: 0.397777
epoch 16; iter: 200; batch classifier loss: 0.424673
epoch 17; iter: 0; batch classifier loss: 0.391509
epoch 17; iter: 200; batch classifier loss: 0.425758
epoch 18; iter: 0; batch classifier loss: 0.477724
epoch 18; iter: 200; batch classifier loss: 0.387291
epoch 19; iter: 0; batch classifier loss: 0.457030
epoch 19; iter: 200; batch classifier loss: 0.417751
epoch 20; iter: 0; batch classifier loss: 0.441797
epoch 20; iter: 200; batch classifier loss: 0.405764
epoch 21; iter: 0; batch classifier loss: 0.405428
epoch 21; iter: 200; batch classifier loss: 0.403910
epoch 22; iter: 0; batch classifier loss: 0.375224
epoch 22; iter: 200; batch classifier loss: 0.419673
epoch 23; iter: 0; batch classifier loss: 0.533660
epoch 23; iter: 200; batch classifier loss: 0.371291
epoch 24; iter: 0; batch classifier loss: 0.406828
epoch 24; iter: 200; batch classifier loss: 0.488163
epoch 25; iter: 0; batch classifier loss: 0.346103
epoch 25; iter: 200; batch classifier loss: 0.452395
epoch 26; iter: 0; batch classifier loss: 0.412668
epoch 26; iter: 200; batch classifier loss: 0.363631
epoch 27; iter: 0; batch classifier loss: 0.402389
epoch 27; iter: 200; batch classifier loss: 0.489910
epoch 28; iter: 0; batch classifier loss: 0.372941
epoch 28; iter: 200; batch classifier loss: 0.486320
epoch 29; iter: 0; batch classifier loss: 0.320363
epoch 29; iter: 200; batch classifier loss: 0.328827
epoch 30; iter: 0; batch classifier loss: 0.365005
epoch 30; iter: 200; batch classifier loss: 0.360580
epoch 31; iter: 0; batch classifier loss: 0.375242
epoch 31; iter: 200; batch classifier loss: 0.438564
epoch 32; iter: 0; batch classifier loss: 0.425088
epoch 32; iter: 200; batch classifier loss: 0.458704
epoch 33; iter: 0; batch classifier loss: 0.337520
epoch 33; iter: 200; batch classifier loss: 0.441541
epoch 34; iter: 0; batch classifier loss: 0.376818
epoch 34; iter: 200; batch classifier loss: 0.364216
epoch 35; iter: 0; batch classifier loss: 0.426272
epoch 35; iter: 200; batch classifier loss: 0.481763
epoch 36; iter: 0; batch classifier loss: 0.456236
epoch 36; iter: 200; batch classifier loss: 0.414621
epoch 37; iter: 0; batch classifier loss: 0.422252
epoch 37; iter: 200; batch classifier loss: 0.374012
epoch 38; iter: 0; batch classifier loss: 0.361711
epoch 38; iter: 200; batch classifier loss: 0.452972
epoch 39; iter: 0; batch classifier loss: 0.489286
epoch 39; iter: 200; batch classifier loss: 0.402996
epoch 40; iter: 0; batch classifier loss: 0.391851
epoch 40; iter: 200; batch classifier loss: 0.416570
epoch 41; iter: 0; batch classifier loss: 0.419140
epoch 41; iter: 200; batch classifier loss: 0.438240
epoch 42; iter: 0; batch classifier loss: 0.349720
epoch 42; iter: 200; batch classifier loss: 0.451622
epoch 43; iter: 0; batch classifier loss: 0.421146
epoch 43; iter: 200; batch classifier loss: 0.464925
epoch 44; iter: 0; batch classifier loss: 0.491823
epoch 44; iter: 200; batch classifier loss: 0.411421
epoch 45; iter: 0; batch classifier loss: 0.472126
epoch 45; iter: 200; batch classifier loss: 0.336557
epoch 46; iter: 0; batch classifier loss: 0.486344
epoch 46; iter: 200; batch classifier loss: 0.452862
epoch 47; iter: 0; batch classifier loss: 0.376278
epoch 47; iter: 200; batch classifier loss: 0.320042
epoch 48; iter: 0; batch classifier loss: 0.484151
epoch 48; iter: 200; batch classifier loss: 0.463232
epoch 49; iter: 0; batch classifier loss: 0.409596
epoch 49; iter: 200; batch classifier loss: 0.445260
```

[11]:
```
<aif360.algorithms.inprocessing.adversarial_debiasing.AdversarialDebiasing at
0x7feab461ba60>
```

[12]:
```
# Apply the plain model to test data
dataset_nodebiasing_train = plain_model.predict(dataset_orig_train)
dataset_nodebiasing_test = plain_model.predict(dataset_orig_test)
```

[13]:
```
# Metrics for the dataset from plain model (without debiasing)
display(Markdown("#### Plain model - without debiasing - dataset metrics"))
metric_dataset_nodebiasing_train = ␣
 ↪BinaryLabelDatasetMetric(dataset_nodebiasing_train,
                                                    ␣
 ↪unprivileged_groups=unprivileged_groups,
                                                    ␣
 ↪privileged_groups=privileged_groups)

print("Train set: Difference in mean outcomes between unprivileged and␣
 ↪privileged groups = %f" % metric_dataset_nodebiasing_train.mean_difference())

metric_dataset_nodebiasing_test = ␣
 ↪BinaryLabelDatasetMetric(dataset_nodebiasing_test,
                                                    ␣
 ↪unprivileged_groups=unprivileged_groups,
                                                    ␣
 ↪privileged_groups=privileged_groups)

print("Test set: Difference in mean outcomes between unprivileged and␣
 ↪privileged groups = %f" % metric_dataset_nodebiasing_test.mean_difference())

display(Markdown("#### Plain model - without debiasing - classification␣
 ↪metrics"))
classified_metric_nodebiasing_test = ClassificationMetric(dataset_orig_test,
                                                 dataset_nodebiasing_test,
                                                ␣
 ↪unprivileged_groups=unprivileged_groups,
                                                ␣
 ↪privileged_groups=privileged_groups)
print("Test set: Classification accuracy = %f" %␣
 ↪classified_metric_nodebiasing_test.accuracy())
print("Test set: Classification precision = %f" %␣
 ↪classified_metric_nodebiasing_test.precision())
print("Test set: Classification recall = %f" %␣
 ↪classified_metric_nodebiasing_test.recall())
```

```
TPR = classified_metric_nodebiasing_test.true_positive_rate()
TNR = classified_metric_nodebiasing_test.true_negative_rate()
FPR = classified_metric_nodebiasing_test.false_positive_rate()
FNR = classified_metric_nodebiasing_test.false_negative_rate()
print("TPR: True Positive Rate = %f" % TPR)
print("TNR: True Negative Rate = %f" % TNR)
print("FPR: False Positive Rate = %f" % FPR)
print("FNR: False Negative Rate = %f" % FNR)
bal_acc_nodebiasing_test = 0.5*(TPR+TNR)
print("Test set: Balanced classification accuracy = %f" %␣
  ↪bal_acc_nodebiasing_test)
print("Test set: Disparate impact = %f" % classified_metric_nodebiasing_test.
  ↪disparate_impact())
print("Test set: Equal opportunity difference = %f" %␣
  ↪classified_metric_nodebiasing_test.equal_opportunity_difference())
print("Test set: Average odds difference = %f" %␣
  ↪classified_metric_nodebiasing_test.average_odds_difference())
print("Test set: Theil_index = %f" % classified_metric_nodebiasing_test.
  ↪theil_index())
```

Plain model - without debiasing - dataset metrics

```
Train set: Difference in mean outcomes between unprivileged and privileged
groups = -0.209740
Test set: Difference in mean outcomes between unprivileged and privileged groups
= -0.214823
```

Plain model - without debiasing - classification metrics

```
Test set: Classification accuracy = 0.798403
Test set: Classification precision = 0.656886
Test set: Classification recall = 0.383743
TPR: True Positive Rate = 0.383743
TNR: True Negative Rate = 0.934306
FPR: False Positive Rate = 0.065694
FNR: False Negative Rate = 0.616257
Test set: Balanced classification accuracy = 0.659025
Test set: Disparate impact = 0.000000
Test set: Equal opportunity difference = -0.451235
Test set: Average odds difference = -0.279242
Test set: Theil_index = 0.184736
```

3.0.2 Apply in-processing algorithm based on adversarial learning

[14]:
```
sess.close()
tf.reset_default_graph()
sess = tf.Session()
```

[15]:
```
# Learn parameters with debias set to True
debiased_model = AdversarialDebiasing(privileged_groups = privileged_groups,
                          unprivileged_groups = unprivileged_groups,
                          scope_name='debiased_classifier_part_1',
                          debias=True,
                          sess=sess)
```

[16]:
```
debiased_model.fit(dataset_orig_train)
```

```
epoch 0; iter: 0; batch classifier loss: 0.759828; batch adversarial loss:
0.661721
epoch 0; iter: 200; batch classifier loss: 0.543180; batch adversarial loss:
0.649956
epoch 1; iter: 0; batch classifier loss: 0.491863; batch adversarial loss:
0.682094
epoch 1; iter: 200; batch classifier loss: 0.459154; batch adversarial loss:
0.649397
epoch 2; iter: 0; batch classifier loss: 0.490993; batch adversarial loss:
0.623625
epoch 2; iter: 200; batch classifier loss: 0.464788; batch adversarial loss:
0.646605
epoch 3; iter: 0; batch classifier loss: 0.444911; batch adversarial loss:
0.641664
epoch 3; iter: 200; batch classifier loss: 0.454854; batch adversarial loss:
0.589226
epoch 4; iter: 0; batch classifier loss: 0.409471; batch adversarial loss:
0.563411
epoch 4; iter: 200; batch classifier loss: 0.408660; batch adversarial loss:
0.608686
epoch 5; iter: 0; batch classifier loss: 0.409663; batch adversarial loss:
0.638049
epoch 5; iter: 200; batch classifier loss: 0.423042; batch adversarial loss:
0.633316
epoch 6; iter: 0; batch classifier loss: 0.432792; batch adversarial loss:
0.572669
epoch 6; iter: 200; batch classifier loss: 0.488844; batch adversarial loss:
0.653281
epoch 7; iter: 0; batch classifier loss: 0.456157; batch adversarial loss:
0.683162
epoch 7; iter: 200; batch classifier loss: 0.388410; batch adversarial loss:
0.584902
epoch 8; iter: 0; batch classifier loss: 0.392698; batch adversarial loss:
```

```
0.598090
epoch 8; iter: 200; batch classifier loss: 0.427080; batch adversarial loss:
0.622040
epoch 9; iter: 0; batch classifier loss: 0.330772; batch adversarial loss:
0.587082
epoch 9; iter: 200; batch classifier loss: 0.441107; batch adversarial loss:
0.642647
epoch 10; iter: 0; batch classifier loss: 0.516407; batch adversarial loss:
0.639106
epoch 10; iter: 200; batch classifier loss: 0.400474; batch adversarial loss:
0.640723
epoch 11; iter: 0; batch classifier loss: 0.386546; batch adversarial loss:
0.606838
epoch 11; iter: 200; batch classifier loss: 0.425216; batch adversarial loss:
0.599930
epoch 12; iter: 0; batch classifier loss: 0.359040; batch adversarial loss:
0.581891
epoch 12; iter: 200; batch classifier loss: 0.474436; batch adversarial loss:
0.588787
epoch 13; iter: 0; batch classifier loss: 0.421059; batch adversarial loss:
0.640704
epoch 13; iter: 200; batch classifier loss: 0.418559; batch adversarial loss:
0.630057
epoch 14; iter: 0; batch classifier loss: 0.407413; batch adversarial loss:
0.642177
epoch 14; iter: 200; batch classifier loss: 0.424837; batch adversarial loss:
0.624600
epoch 15; iter: 0; batch classifier loss: 0.442553; batch adversarial loss:
0.613652
epoch 15; iter: 200; batch classifier loss: 0.430757; batch adversarial loss:
0.591305
epoch 16; iter: 0; batch classifier loss: 0.413538; batch adversarial loss:
0.638950
epoch 16; iter: 200; batch classifier loss: 0.480594; batch adversarial loss:
0.554091
epoch 17; iter: 0; batch classifier loss: 0.389297; batch adversarial loss:
0.645739
epoch 17; iter: 200; batch classifier loss: 0.388353; batch adversarial loss:
0.609963
epoch 18; iter: 0; batch classifier loss: 0.482714; batch adversarial loss:
0.652097
epoch 18; iter: 200; batch classifier loss: 0.360246; batch adversarial loss:
0.607083
epoch 19; iter: 0; batch classifier loss: 0.414243; batch adversarial loss:
0.612275
epoch 19; iter: 200; batch classifier loss: 0.428737; batch adversarial loss:
0.569746
epoch 20; iter: 0; batch classifier loss: 0.514808; batch adversarial loss:
```

```
0.577729
epoch 20; iter: 200; batch classifier loss: 0.458264; batch adversarial loss:
0.573793
epoch 21; iter: 0; batch classifier loss: 0.450629; batch adversarial loss:
0.593602
epoch 21; iter: 200; batch classifier loss: 0.392465; batch adversarial loss:
0.568886
epoch 22; iter: 0; batch classifier loss: 0.401530; batch adversarial loss:
0.569516
epoch 22; iter: 200; batch classifier loss: 0.485607; batch adversarial loss:
0.600458
epoch 23; iter: 0; batch classifier loss: 0.379045; batch adversarial loss:
0.600398
epoch 23; iter: 200; batch classifier loss: 0.408872; batch adversarial loss:
0.593203
epoch 24; iter: 0; batch classifier loss: 0.411928; batch adversarial loss:
0.628431
epoch 24; iter: 200; batch classifier loss: 0.394602; batch adversarial loss:
0.611689
epoch 25; iter: 0; batch classifier loss: 0.442638; batch adversarial loss:
0.607520
epoch 25; iter: 200; batch classifier loss: 0.444454; batch adversarial loss:
0.581680
epoch 26; iter: 0; batch classifier loss: 0.360738; batch adversarial loss:
0.607105
epoch 26; iter: 200; batch classifier loss: 0.427161; batch adversarial loss:
0.623552
epoch 27; iter: 0; batch classifier loss: 0.394361; batch adversarial loss:
0.606730
epoch 27; iter: 200; batch classifier loss: 0.420072; batch adversarial loss:
0.542470
epoch 28; iter: 0; batch classifier loss: 0.352984; batch adversarial loss:
0.636810
epoch 28; iter: 200; batch classifier loss: 0.380665; batch adversarial loss:
0.635741
epoch 29; iter: 0; batch classifier loss: 0.497517; batch adversarial loss:
0.661010
epoch 29; iter: 200; batch classifier loss: 0.383153; batch adversarial loss:
0.575549
epoch 30; iter: 0; batch classifier loss: 0.441408; batch adversarial loss:
0.637900
epoch 30; iter: 200; batch classifier loss: 0.420395; batch adversarial loss:
0.601642
epoch 31; iter: 0; batch classifier loss: 0.437370; batch adversarial loss:
0.585287
epoch 31; iter: 200; batch classifier loss: 0.432868; batch adversarial loss:
0.604522
epoch 32; iter: 0; batch classifier loss: 0.481593; batch adversarial loss:
```

```
0.583065
epoch 32; iter: 200; batch classifier loss: 0.404837; batch adversarial loss:
0.574579
epoch 33; iter: 0; batch classifier loss: 0.438782; batch adversarial loss:
0.599505
epoch 33; iter: 200; batch classifier loss: 0.374081; batch adversarial loss:
0.642765
epoch 34; iter: 0; batch classifier loss: 0.477244; batch adversarial loss:
0.611955
epoch 34; iter: 200; batch classifier loss: 0.391119; batch adversarial loss:
0.570449
epoch 35; iter: 0; batch classifier loss: 0.464945; batch adversarial loss:
0.621999
epoch 35; iter: 200; batch classifier loss: 0.455226; batch adversarial loss:
0.732162
epoch 36; iter: 0; batch classifier loss: 0.429898; batch adversarial loss:
0.555084
epoch 36; iter: 200; batch classifier loss: 0.504493; batch adversarial loss:
0.588231
epoch 37; iter: 0; batch classifier loss: 0.282065; batch adversarial loss:
0.628801
epoch 37; iter: 200; batch classifier loss: 0.419385; batch adversarial loss:
0.599433
epoch 38; iter: 0; batch classifier loss: 0.409219; batch adversarial loss:
0.610933
epoch 38; iter: 200; batch classifier loss: 0.378450; batch adversarial loss:
0.604570
epoch 39; iter: 0; batch classifier loss: 0.353169; batch adversarial loss:
0.685699
epoch 39; iter: 200; batch classifier loss: 0.577848; batch adversarial loss:
0.611851
epoch 40; iter: 0; batch classifier loss: 0.307270; batch adversarial loss:
0.640353
epoch 40; iter: 200; batch classifier loss: 0.492280; batch adversarial loss:
0.566908
epoch 41; iter: 0; batch classifier loss: 0.425078; batch adversarial loss:
0.603693
epoch 41; iter: 200; batch classifier loss: 0.450408; batch adversarial loss:
0.646848
epoch 42; iter: 0; batch classifier loss: 0.434968; batch adversarial loss:
0.530884
epoch 42; iter: 200; batch classifier loss: 0.438363; batch adversarial loss:
0.600380
epoch 43; iter: 0; batch classifier loss: 0.407435; batch adversarial loss:
0.627726
epoch 43; iter: 200; batch classifier loss: 0.430631; batch adversarial loss:
0.678400
epoch 44; iter: 0; batch classifier loss: 0.414942; batch adversarial loss:
```

```
0.615971
epoch 44; iter: 200; batch classifier loss: 0.444667; batch adversarial loss:
0.605705
epoch 45; iter: 0; batch classifier loss: 0.405813; batch adversarial loss:
0.615326
epoch 45; iter: 200; batch classifier loss: 0.429467; batch adversarial loss:
0.566638
epoch 46; iter: 0; batch classifier loss: 0.305157; batch adversarial loss:
0.592450
epoch 46; iter: 200; batch classifier loss: 0.458519; batch adversarial loss:
0.592918
epoch 47; iter: 0; batch classifier loss: 0.511870; batch adversarial loss:
0.561661
epoch 47; iter: 200; batch classifier loss: 0.523474; batch adversarial loss:
0.590973
epoch 48; iter: 0; batch classifier loss: 0.436339; batch adversarial loss:
0.613237
epoch 48; iter: 200; batch classifier loss: 0.429411; batch adversarial loss:
0.644450
epoch 49; iter: 0; batch classifier loss: 0.455125; batch adversarial loss:
0.585150
epoch 49; iter: 200; batch classifier loss: 0.457503; batch adversarial loss:
0.528493
```

[16]:
```
<aif360.algorithms.inprocessing.adversarial_debiasing.AdversarialDebiasing at
0x7feab41df790>
```

[17]:
```python
# Apply the plain model to test data
dataset_debiasing_train = debiased_model.predict(dataset_orig_train)
dataset_debiasing_test = debiased_model.predict(dataset_orig_test)
```

[18]:
```python
# Metrics for the dataset from plain model (without debiasing)
display(Markdown("#### Plain model - without debiasing - dataset metrics"))
print("Train set: Difference in mean outcomes between unprivileged and␣
  ↪privileged groups = %f" % metric_dataset_nodebiasing_train.mean_difference())
print("Test set: Difference in mean outcomes between unprivileged and␣
  ↪privileged groups = %f" % metric_dataset_nodebiasing_test.mean_difference())

# Metrics for the dataset from model with debiasing
display(Markdown("#### Model - with debiasing - dataset metrics"))
metric_dataset_debiasing_train =␣
  ↪BinaryLabelDatasetMetric(dataset_debiasing_train,
                                                               ␣
  ↪unprivileged_groups=unprivileged_groups,
                                                               ␣
  ↪privileged_groups=privileged_groups)
```

```
print("Train set: Difference in mean outcomes between unprivileged and␣
↪privileged groups = %f" % metric_dataset_debiasing_train.mean_difference())

metric_dataset_debiasing_test = BinaryLabelDatasetMetric(dataset_debiasing_test,
                                                         ␣
↪unprivileged_groups=unprivileged_groups,
                                                         ␣
↪privileged_groups=privileged_groups)

print("Test set: Difference in mean outcomes between unprivileged and␣
↪privileged groups = %f" % metric_dataset_debiasing_test.mean_difference())



display(Markdown("#### Plain model - without debiasing - classification␣
↪metrics"))
print("Test set: Classification accuracy = %f" %␣
↪classified_metric_nodebiasing_test.accuracy())
print("Test set: Classification precision = %f" %␣
↪classified_metric_nodebiasing_test.precision())
print("Test set: Classification recall = %f" %␣
↪classified_metric_nodebiasing_test.recall())
TPR = classified_metric_nodebiasing_test.true_positive_rate()
TNR = classified_metric_nodebiasing_test.true_negative_rate()
FPR = classified_metric_nodebiasing_test.false_positive_rate()
FNR = classified_metric_nodebiasing_test.false_negative_rate()
print("TPR: True Positive Rate = %f" % TPR)
print("TNR: True Negative Rate = %f" % TNR)
print("FPR: False Positive Rate = %f" % FPR)
print("FNR: False Negative Rate = %f" % FNR)
bal_acc_nodebiasing_test = 0.5*(TPR+TNR)
print("Test set: Balanced classification accuracy = %f" %␣
↪bal_acc_nodebiasing_test)
print("Test set: Disparate impact = %f" % classified_metric_nodebiasing_test.
↪disparate_impact())
print("Test set: Equal opportunity difference = %f" %␣
↪classified_metric_nodebiasing_test.equal_opportunity_difference())
print("Test set: Average odds difference = %f" %␣
↪classified_metric_nodebiasing_test.average_odds_difference())
print("Test set: Theil_index = %f" % classified_metric_nodebiasing_test.
↪theil_index())



display(Markdown("#### Model - with debiasing - classification metrics"))
classified_metric_debiasing_test = ClassificationMetric(dataset_orig_test,
                                                    dataset_debiasing_test,
                                                  ␣
↪unprivileged_groups=unprivileged_groups,
                                                  ␣
↪privileged_groups=privileged_groups)
print("Test set: Classification accuracy = %f" %␣
↪classified_metric_debiasing_test.accuracy())
TPR = classified_metric_nodebiasing_test.true_positive_rate()
TNR = classified_metric_nodebiasing_test.true_negative_rate()
FPR = classified_metric_nodebiasing_test.false_positive_rate()
FNR = classified_metric_nodebiasing_test.false_negative_rate()
print("TPR: True Positive Rate = %f" % TPR)
print("TNR: True Negative Rate = %f" % TNR)
print("FPR: False Positive Rate = %f" % FPR)
print("FNR: False Negative Rate = %f" % FNR)
bal_acc_debiasing_test = 0.5*(TPR+TNR)
print("Test set: Balanced classification accuracy = %f" %␣
↪bal_acc_debiasing_test)
print("Test set: Disparate impact = %f" % classified_metric_debiasing_test.
↪disparate_impact())
print("Test set: Equal opportunity difference = %f" %␣
↪classified_metric_debiasing_test.equal_opportunity_difference())
print("Test set: Average odds difference = %f" %␣
↪classified_metric_debiasing_test.average_odds_difference())
print("Test set: Theil_index = %f" % classified_metric_debiasing_test.
↪theil_index())
```

Plain model - without debiasing - dataset metrics

```
Train set: Difference in mean outcomes between unprivileged and privileged
groups = -0.209740
Test set: Difference in mean outcomes between unprivileged and privileged groups
= -0.214823
```

Model - with debiasing - dataset metrics

```
Train set: Difference in mean outcomes between unprivileged and privileged
groups = -0.136748
Test set: Difference in mean outcomes between unprivileged and privileged groups
= -0.142406
```

Plain model - without debiasing - classification metrics

```
Test set: Classification accuracy = 0.798403
Test set: Classification precision = 0.656886
Test set: Classification recall = 0.383743
TPR: True Positive Rate = 0.383743
TNR: True Negative Rate = 0.934306
```

```
FPR: False Positive Rate = 0.065694
FNR: False Negative Rate = 0.616257
Test set: Balanced classification accuracy = 0.659025
Test set: Disparate impact = 0.000000
Test set: Equal opportunity difference = -0.451235
Test set: Average odds difference = -0.279242
Test set: Theil_index = 0.184736
```

Model - with debiasing - classification metrics

```
Test set: Classification accuracy = 0.791169
TPR: True Positive Rate = 0.383743
TNR: True Negative Rate = 0.934306
FPR: False Positive Rate = 0.065694
FNR: False Negative Rate = 0.616257
Test set: Balanced classification accuracy = 0.659025
Test set: Disparate impact = 0.316061
Test set: Equal opportunity difference = -0.244574
Test set: Average odds difference = -0.148872
Test set: Theil_index = 0.182832
```

4 Part 2: Preprocessing dataset using MinMaxScaler

StandardScaler also gives approximately the same results.

5 Part 3: changing the architecture of the neural network, added 2 additonal layer with the activation function relu

[19]:
```
from aif360.algorithms.inprocessing.adversarial_debiasing_pg1 import␣
  ↪AdversarialDebiasing
```

[20]:
```
# Add more data preprocessing here
min_max_scaler = MinMaxScaler()
dataset_orig_train.features = min_max_scaler.fit_transform(dataset_orig_train.
  ↪features)
dataset_orig_test.features = min_max_scaler.transform(dataset_orig_test.
  ↪features)
metric_scaled_train = BinaryLabelDatasetMetric(dataset_orig_train,
                             unprivileged_groups=unprivileged_groups,
                             privileged_groups=privileged_groups)
display(Markdown("#### Scaled dataset - Verify that the scaling does not affect␣
  ↪the group label statistics"))
print("Train set: Difference in mean outcomes between unprivileged and␣
  ↪privileged groups = %f" % metric_scaled_train.mean_difference())
metric_scaled_test = BinaryLabelDatasetMetric(dataset_orig_test,
                             unprivileged_groups=unprivileged_groups,
                               privileged_groups=privileged_groups)
print("Test set: Difference in mean outcomes between unprivileged and␣
 ↪privileged groups = %f" % metric_scaled_test.mean_difference())
```

Scaled dataset - Verify that the scaling does not affect the group label statistics

```
Train set: Difference in mean outcomes between unprivileged and privileged
groups = -0.191948
Test set: Difference in mean outcomes between unprivileged and privileged groups
= -0.200418
```

5.0.1 Learn plain classifier without debiasing

[21]:
```
# Load post-processing algorithm that equalizes the odds
# Learn parameters with debias set to False
sess = tf.Session()
plain_model = AdversarialDebiasing(privileged_groups = privileged_groups,
                          unprivileged_groups = unprivileged_groups,
                          scope_name='plain_classifier_part_3',
                          debias=False,
                          sess=sess)
```

[22]:
```
plain_model.fit(dataset_orig_train)
```

```
epoch 0; iter: 0; batch classifier loss: 0.687325
epoch 0; iter: 200; batch classifier loss: 0.470102
epoch 1; iter: 0; batch classifier loss: 0.550684
epoch 1; iter: 200; batch classifier loss: 0.497577
epoch 2; iter: 0; batch classifier loss: 0.397570
epoch 2; iter: 200; batch classifier loss: 0.399183
epoch 3; iter: 0; batch classifier loss: 0.416078
epoch 3; iter: 200; batch classifier loss: 0.402656
epoch 4; iter: 0; batch classifier loss: 0.422276
epoch 4; iter: 200; batch classifier loss: 0.425741
epoch 5; iter: 0; batch classifier loss: 0.457593
epoch 5; iter: 200; batch classifier loss: 0.377042
epoch 6; iter: 0; batch classifier loss: 0.515300
epoch 6; iter: 200; batch classifier loss: 0.409753
epoch 7; iter: 0; batch classifier loss: 0.427897
epoch 7; iter: 200; batch classifier loss: 0.430777
epoch 8; iter: 0; batch classifier loss: 0.376762
epoch 8; iter: 200; batch classifier loss: 0.433482
epoch 9; iter: 0; batch classifier loss: 0.465370
epoch 9; iter: 200; batch classifier loss: 0.446604
epoch 10; iter: 0; batch classifier loss: 0.375945
epoch 10; iter: 200; batch classifier loss: 0.734741
epoch 11; iter: 0; batch classifier loss: 0.387232
epoch 11; iter: 200; batch classifier loss: 65.986748
epoch 12; iter: 0; batch classifier loss: 68.001854
epoch 12; iter: 200; batch classifier loss: 31.945459
epoch 13; iter: 0; batch classifier loss: 9.398496
epoch 13; iter: 200; batch classifier loss: 6.801931
epoch 14; iter: 0; batch classifier loss: 14.070862
epoch 14; iter: 200; batch classifier loss: 13.970785
epoch 15; iter: 0; batch classifier loss: 15.594452
epoch 15; iter: 200; batch classifier loss: 17.829063
epoch 16; iter: 0; batch classifier loss: 15.732504
epoch 16; iter: 200; batch classifier loss: 16.248775
epoch 17; iter: 0; batch classifier loss: 5.934124
epoch 17; iter: 200; batch classifier loss: 3.371675
epoch 18; iter: 0; batch classifier loss: 18.518974
epoch 18; iter: 200; batch classifier loss: 14.403069
epoch 19; iter: 0; batch classifier loss: 10.671350
epoch 19; iter: 200; batch classifier loss: 2.919434
epoch 20; iter: 0; batch classifier loss: 8.843431
epoch 20; iter: 200; batch classifier loss: 14.541245
epoch 21; iter: 0; batch classifier loss: 11.543805
epoch 21; iter: 200; batch classifier loss: 3.560473
epoch 22; iter: 0; batch classifier loss: 4.540210
epoch 22; iter: 200; batch classifier loss: 5.168231
epoch 23; iter: 0; batch classifier loss: 2.040995
epoch 23; iter: 200; batch classifier loss: 2.927728
epoch 24; iter: 0; batch classifier loss: 2.587284
epoch 24; iter: 200; batch classifier loss: 2.208058
epoch 25; iter: 0; batch classifier loss: 4.067185
epoch 25; iter: 200; batch classifier loss: 3.760564
epoch 26; iter: 0; batch classifier loss: 2.659660
epoch 26; iter: 200; batch classifier loss: 3.721421
epoch 27; iter: 0; batch classifier loss: 2.257256
epoch 27; iter: 200; batch classifier loss: 3.395134
epoch 28; iter: 0; batch classifier loss: 2.745183
epoch 28; iter: 200; batch classifier loss: 2.259810
epoch 29; iter: 0; batch classifier loss: 3.626838
epoch 29; iter: 200; batch classifier loss: 4.520951
epoch 30; iter: 0; batch classifier loss: 3.177800
epoch 30; iter: 200; batch classifier loss: 3.354076
epoch 31; iter: 0; batch classifier loss: 1.657869
epoch 31; iter: 200; batch classifier loss: 1.294822
epoch 32; iter: 0; batch classifier loss: 3.941934
epoch 32; iter: 200; batch classifier loss: 2.551117
epoch 33; iter: 0; batch classifier loss: 2.395140
epoch 33; iter: 200; batch classifier loss: 0.711682
epoch 34; iter: 0; batch classifier loss: 1.167344
epoch 34; iter: 200; batch classifier loss: 6.624094
epoch 35; iter: 0; batch classifier loss: 1.596829
epoch 35; iter: 200; batch classifier loss: 0.860357
epoch 36; iter: 0; batch classifier loss: 3.961234
epoch 36; iter: 200; batch classifier loss: 0.763135
epoch 37; iter: 0; batch classifier loss: 0.550988
epoch 37; iter: 200; batch classifier loss: 3.709104
epoch 38; iter: 0; batch classifier loss: 3.750732
epoch 38; iter: 200; batch classifier loss: 0.422408
epoch 39; iter: 0; batch classifier loss: 0.823567
epoch 39; iter: 200; batch classifier loss: 0.960444
epoch 40; iter: 0; batch classifier loss: 16.036165
epoch 40; iter: 200; batch classifier loss: 9.790880
epoch 41; iter: 0; batch classifier loss: 5.315099
epoch 41; iter: 200; batch classifier loss: 4.862730
epoch 42; iter: 0; batch classifier loss: 2.565994
epoch 42; iter: 200; batch classifier loss: 4.369214
epoch 43; iter: 0; batch classifier loss: 1.520716
epoch 43; iter: 200; batch classifier loss: 2.605941
epoch 44; iter: 0; batch classifier loss: 1.412490
epoch 44; iter: 200; batch classifier loss: 1.860626
epoch 45; iter: 0; batch classifier loss: 1.120535
epoch 45; iter: 200; batch classifier loss: 1.669161
epoch 46; iter: 0; batch classifier loss: 1.128938
epoch 46; iter: 200; batch classifier loss: 1.341059
epoch 47; iter: 0; batch classifier loss: 2.550440
epoch 47; iter: 200; batch classifier loss: 0.803675
epoch 48; iter: 0; batch classifier loss: 1.442381
epoch 48; iter: 200; batch classifier loss: 1.527347
epoch 49; iter: 0; batch classifier loss: 0.690433
epoch 49; iter: 200; batch classifier loss: 1.058060
```

[22]:
```
<aif360.algorithms.inprocessing.adversarial_debiasing_pg1.AdversarialDebiasing
at 0x7feaa0234ac0>
```

[23]:
```python
# Apply the plain model to test data
dataset_nodebiasing_train = plain_model.predict(dataset_orig_train)
dataset_nodebiasing_test = plain_model.predict(dataset_orig_test)
```

[24]:
```python
# Metrics for the dataset from plain model (without debiasing)
display(Markdown("#### Plain model - without debiasing - dataset metrics"))
metric_dataset_nodebiasing_train =␣
 ↪BinaryLabelDatasetMetric(dataset_nodebiasing_train,
                                                       ␣
 ↪unprivileged_groups=unprivileged_groups,
                                                       ␣
 ↪privileged_groups=privileged_groups)
```

```
print("Train set: Difference in mean outcomes between unprivileged and␣
  ↪privileged groups = %f" % metric_dataset_nodebiasing_train.mean_difference())

metric_dataset_nodebiasing_test =␣
  ↪BinaryLabelDatasetMetric(dataset_nodebiasing_test,
                                                                           ␣
  ↪unprivileged_groups=unprivileged_groups,
                                                                           ␣
  ↪privileged_groups=privileged_groups)

print("Test set: Difference in mean outcomes between unprivileged and␣
  ↪privileged groups = %f" % metric_dataset_nodebiasing_test.mean_difference())

display(Markdown("#### Plain model - without debiasing - classification␣
  ↪metrics"))
classified_metric_nodebiasing_test = ClassificationMetric(dataset_orig_test,
                                                 dataset_nodebiasing_test,
                                                                               ␣
  ↪unprivileged_groups=unprivileged_groups,
                                                                               ␣
  ↪privileged_groups=privileged_groups)
print("Test set: Classification accuracy = %f" %␣
  ↪classified_metric_nodebiasing_test.accuracy())
print("Test set: Classification precision = %f" %␣
  ↪classified_metric_nodebiasing_test.precision())
print("Test set: Classification recall = %f" %␣
  ↪classified_metric_nodebiasing_test.recall())
TPR = classified_metric_nodebiasing_test.true_positive_rate()
TNR = classified_metric_nodebiasing_test.true_negative_rate()
FPR = classified_metric_nodebiasing_test.false_positive_rate()
FNR = classified_metric_nodebiasing_test.false_negative_rate()
print("TPR: True Positive Rate = %f" % TPR)
print("TNR: True Negative Rate = %f" % TNR)
print("FPR: False Positive Rate = %f" % FPR)
print("FNR: False Negative Rate = %f" % FNR)
bal_acc_nodebiasing_test = 0.5*(TPR+TNR)
print("Test set: Balanced classification accuracy = %f" %␣
  ↪bal_acc_nodebiasing_test)
print("Test set: Disparate impact = %f" % classified_metric_nodebiasing_test.
  ↪disparate_impact())
print("Test set: Equal opportunity difference = %f" %␣
  ↪classified_metric_nodebiasing_test.equal_opportunity_difference())
print("Test set: Average odds difference = %f" %␣
  ↪classified_metric_nodebiasing_test.average_odds_difference())
print("Test set: Theil_index = %f" % classified_metric_nodebiasing_test.
  ↪theil_index())
```

Plain model - without debiasing - dataset metrics

```
Train set: Difference in mean outcomes between unprivileged and privileged
groups = 0.000000
Test set: Difference in mean outcomes between unprivileged and privileged groups
= 0.000000
```

Plain model - without debiasing - classification metrics

```
Test set: Classification accuracy = 0.753156
Test set: Classification precision = 0.000000
Test set: Classification recall = 0.000000
TPR: True Positive Rate = 0.000000
TNR: True Negative Rate = 1.000000
FPR: False Positive Rate = 0.000000
FNR: False Negative Rate = 1.000000
Test set: Balanced classification accuracy = 0.500000
Test set: Disparate impact = nan
Test set: Equal opportunity difference = 0.000000
Test set: Average odds difference = 0.000000
Test set: Theil_index = 0.283482
```

```
invalid value encountered in double_scalars
```

5.0.2 Apply in-processing algorithm based on adversarial learning

[25]:
```
sess.close()
tf.reset_default_graph()
sess = tf.Session()
```

[26]:
```
# Learn parameters with debias set to True
debiased_model = AdversarialDebiasing(privileged_groups = privileged_groups,
                          unprivileged_groups = unprivileged_groups,
                          scope_name='debiased_classifier_part_2',
                          debias=True,
                          sess=sess)
```

[27]:
```
debiased_model.fit(dataset_orig_train)
```

```
epoch 0; iter: 0; batch classifier loss: 0.681401; batch adversarial loss:
0.649939
epoch 0; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 1; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 1; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 2; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 2; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 3; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 3; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 4; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 4; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 5; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 5; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 6; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 6; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 7; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 7; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 8; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 8; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 9; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 9; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 10; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 10; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 11; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 11; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 12; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 12; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 13; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 13; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 14; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 14; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 15; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 15; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 16; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 16; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 17; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 17; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 18; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 18; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 19; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 19; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 20; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 20; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 21; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 21; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 22; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 22; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 23; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 23; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 24; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 24; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 25; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 25; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 26; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 26; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 27; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 27; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 28; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 28; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 29; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 29; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 30; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 30; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 31; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 31; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 32; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 32; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 33; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 33; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 34; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 34; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 35; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 35; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 36; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 36; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 37; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 37; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 38; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 38; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 39; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 39; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 40; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 40; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 41; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 41; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 42; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 42; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 43; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 43; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 44; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 44; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 45; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 45; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 46; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 46; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 47; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 47; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 48; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 48; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
epoch 49; iter: 0; batch classifier loss: nan; batch adversarial loss: nan
epoch 49; iter: 200; batch classifier loss: nan; batch adversarial loss: nan
```

```
[27]: <aif360.algorithms.inprocessing.adversarial_debiasing_pg1.AdversarialDebiasing
      at 0x7feaa02cef40>
```

[28]:
```
# Apply the plain model to test data
dataset_debiasing_train = debiased_model.predict(dataset_orig_train)
dataset_debiasing_test = debiased_model.predict(dataset_orig_test)
```

[29]:
```
# Metrics for the dataset from plain model (without debiasing)
display(Markdown("#### Plain model - without debiasing - dataset metrics"))
print("Train set: Difference in mean outcomes between unprivileged and␣
 ↪privileged groups = %f" % metric_dataset_nodebiasing_train.mean_difference())
print("Test set: Difference in mean outcomes between unprivileged and␣
 ↪privileged groups = %f" % metric_dataset_nodebiasing_test.mean_difference())

# Metrics for the dataset from model with debiasing
display(Markdown("#### Model - with debiasing - dataset metrics"))
metric_dataset_debiasing_train =␣
 ↪BinaryLabelDatasetMetric(dataset_debiasing_train,
                                                     ␣
 ↪unprivileged_groups=unprivileged_groups,
                                                     ␣
 ↪privileged_groups=privileged_groups)

print("Train set: Difference in mean outcomes between unprivileged and␣
 ↪privileged groups = %f" % metric_dataset_debiasing_train.mean_difference())

metric_dataset_debiasing_test = BinaryLabelDatasetMetric(dataset_debiasing_test,
                                                     ␣
 ↪unprivileged_groups=unprivileged_groups,
                                                     ␣
 ↪privileged_groups=privileged_groups)

print("Test set: Difference in mean outcomes between unprivileged and␣
 ↪privileged groups = %f" % metric_dataset_debiasing_test.mean_difference())



display(Markdown("#### Plain model - without debiasing - classification␣
 ↪metrics"))
print("Test set: Classification accuracy = %f" %␣
 ↪classified_metric_nodebiasing_test.accuracy())
print("Test set: Classification precision = %f" %␣
 ↪classified_metric_nodebiasing_test.precision())
print("Test set: Classification recall = %f" %␣
 ↪classified_metric_nodebiasing_test.recall())
TPR = classified_metric_nodebiasing_test.true_positive_rate()
TNR = classified_metric_nodebiasing_test.true_negative_rate()
FPR = classified_metric_nodebiasing_test.false_positive_rate()
FNR = classified_metric_nodebiasing_test.false_negative_rate()
```

```
print("TPR: True Positive Rate = %f" % TPR)
print("TNR: True Negative Rate = %f" % TNR)
print("FPR: False Positive Rate = %f" % FPR)
print("FNR: False Negative Rate = %f" % FNR)
bal_acc_nodebiasing_test = 0.5*(TPR+TNR)
print("Test set: Balanced classification accuracy = %f" %␣
 ↪bal_acc_nodebiasing_test)
print("Test set: Disparate impact = %f" % classified_metric_nodebiasing_test.
 ↪disparate_impact())
print("Test set: Equal opportunity difference = %f" %␣
 ↪classified_metric_nodebiasing_test.equal_opportunity_difference())
print("Test set: Average odds difference = %f" %␣
 ↪classified_metric_nodebiasing_test.average_odds_difference())
print("Test set: Theil_index = %f" % classified_metric_nodebiasing_test.
 ↪theil_index())



display(Markdown("#### Model - with debiasing - classification metrics"))
classified_metric_debiasing_test = ClassificationMetric(dataset_orig_test,
                                                 dataset_debiasing_test,
                                                 ␣
 ↪unprivileged_groups=unprivileged_groups,
                                                 ␣
 ↪privileged_groups=privileged_groups)
print("Test set: Classification accuracy = %f" %␣
 ↪classified_metric_debiasing_test.accuracy())
TPR = classified_metric_nodebiasing_test.true_positive_rate()
TNR = classified_metric_nodebiasing_test.true_negative_rate()
FPR = classified_metric_nodebiasing_test.false_positive_rate()
FNR = classified_metric_nodebiasing_test.false_negative_rate()

print("TPR: True Positive Rate = %f" % TPR)
print("TNR: True Negative Rate = %f" % TNR)
print("FPR: False Positive Rate = %f" % FPR)
print("FNR: False Negative Rate = %f" % FNR)
bal_acc_debiasing_test = 0.5*(TPR+TNR)
print("Test set: Balanced classification accuracy = %f" %␣
 ↪bal_acc_debiasing_test)
print("Test set: Disparate impact = %f" % classified_metric_debiasing_test.
 ↪disparate_impact())
print("Test set: Equal opportunity difference = %f" %␣
 ↪classified_metric_debiasing_test.equal_opportunity_difference())
print("Test set: Average odds difference = %f" %␣
 ↪classified_metric_debiasing_test.average_odds_difference())
```

```
print("Test set: Theil_index = %f" % classified_metric_debiasing_test.
 ↪theil_index())
```

Plain model - without debiasing - dataset metrics

```
Train set: Difference in mean outcomes between unprivileged and privileged
groups = 0.000000
Test set: Difference in mean outcomes between unprivileged and privileged groups
= 0.000000
```

Model - with debiasing - dataset metrics

```
Train set: Difference in mean outcomes between unprivileged and privileged
groups = 0.000000
Test set: Difference in mean outcomes between unprivileged and privileged groups
= 0.000000
```

Plain model - without debiasing - classification metrics

```
Test set: Classification accuracy = 0.753156
Test set: Classification precision = 0.000000
Test set: Classification recall = 0.000000
TPR: True Positive Rate = 0.000000
TNR: True Negative Rate = 1.000000
FPR: False Positive Rate = 0.000000
FNR: False Negative Rate = 1.000000
Test set: Balanced classification accuracy = 0.500000
Test set: Disparate impact = nan
Test set: Equal opportunity difference = 0.000000
Test set: Average odds difference = 0.000000
Test set: Theil_index = 0.283482
```

Model - with debiasing - classification metrics

```
Test set: Classification accuracy = 0.753156
TPR: True Positive Rate = 0.000000
TNR: True Negative Rate = 1.000000
FPR: False Positive Rate = 0.000000
FNR: False Negative Rate = 1.000000
Test set: Balanced classification accuracy = 0.500000
Test set: Disparate impact = nan
Test set: Equal opportunity difference = 0.000000
Test set: Average odds difference = 0.000000
Test set: Theil_index = 0.283482
```

[30]:
```
# #PLOT ROC curve
# # plt.plot(FPR,TPR)
# plt.plot(FPR, TPR, 'k--', label='ROC curve (area = %0.3f)')
# # plt.plot([0, 1], [0, 1], 'k--')  # random predictions curve
# plt.xlim([0.0, 1.0])
# plt.ylim([0.0, 1.0])
# plt.xlabel('False Positive Rate or (1 - Specifity)')
# plt.ylabel('True Positive Rate or (Sensitivity)')
# plt.title('Receiver Operating Characteristic')
# plt.legend(loc="lower right")
# plt.show()

# # from sklearn.metrics import roc_curve
# # from sklearn.metrics import auc
```

References:

[1] B. H. Zhang, B. Lemoine, and M. Mitchell, "Mitigating UnwantedBiases with Adversarial Lear
AAAI/ACM Conference on Artificial Intelligence, Ethics, and Society, 2018.