IBM AI Fairness 360 tool Tutorial

https://aif360.mybluemix.net/

Loading your own dataset

This tutorial is created to make it easier to load your own datasets to the IBM AI fairness 360 tool.

Note1: If you run the Application file (load your own data.ipynb) without changing anything, an example dataset will be loaded.

Note2: In this tutorial, the text in <> brackets are placeholders. It means, when preparing your application, you should change them with text specific to your case.

In order to load your own data to the AI Fairness 360 tool, two files are needed in addition to your application file where you will use AIF360. One with a data load function and one with a class of your dataset.

Application file	**Load function file**	**Data class file**
Format: Notebook **Extension:** .ipynb In this tutorial: • load your own data.ipynb • manage bias by reweighing.ipynb	**Format:** Python class **Extension:** .py In this tutorial: • data_preproc_functions.py	**Format:** Python class **Extension:** .py In this tutorial: • heart_dataset.py
File to include our main actions	Supporting files	

Let's go step by step. You can copy and paste the code to a new notebook or use the ready-to-go notebook that comes with this tutorial. Here is our application file "load your own data.ipynb" which is located in the folder "/code":

Step 1:

Location: Application file (load your own data.ipynb)
Import necessary libraries and files

```
%matplotlib inline

import sys
sys.path.append("../")
import numpy as np
from tqdm import tqdm

from aif360.datasets import BinaryLabelDataset
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 import OptimPreproc

from aif360.algorithms.preprocessing.optim_preproc_helpers.opt_tools import OptTools
# from common_utils import compute_metrics

from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score

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

Step 2:

Location: NA
Create the two supporting files. Name the first one, **data_preproc_functions.py** and the second one **<your dataset name>_dataset.py**

Step 3:

Location: Data class file
Let's work on the data class. Go to **<your dataset name>_dataset.py** and follow the ToDos.
Example class 1, example class 2.

Step 4:

Location: Load function file
Let's work on the load function. Go to **data_preproc_functions.py** and follow the ToDos.
Example load functions.

Step 5:

Location: Application file (load your own data.ipynb)
Import your custom classes and functions by changing the file name and the class name in the import statement.

```
# from <name of your data class file> import <name of the class in the data class file>
from heart_dataset import HeartDiseaseDataset

# from data_preproc_functions import <name of the function in the load function file>
from data_preproc_functions import load_preproc_data_heart
```

Step 6:

Location: Application file (load your own data.ipynb)
Let's set up where we're going to call the load function. Based on your dataset, your protected attribute might be something else such as race, income, gender, etc. you already determined the protected attribute in your data class file. Use the same one. For the rest, follow the ToDos in the file.

```
# import dataset

# TODO 1 - select your protected attribute
protected_attribute = 'sex'
dataset_orig = load_preproc_data_heart([protected_attribute])

# TODO 2 - Based on your understanding determine the privileged and unprivileged groups.
privileged_groups = [{protected_attribute: 1}]
unprivileged_groups = [{protected_attribute: 0}]


#random seed
np.random.seed(1)

# Split into train, validation, and test
dataset_orig_train, dataset_orig_vt = dataset_orig.split([0.7], shuffle=True)
dataset_orig_valid, dataset_orig_test = dataset_orig_vt.split([0.5], shuffle=True)
```

Step 7:

Location: Application file (load your own data.ipynb)
If everything went well, you are ready to load your dataset to your application. Restart the kernel and run all the cells in the notebook in order.

Note3: In order for the changes you made in the supporting files to be recognized, you need to restart the kernel of the notebook and import them again.

If you get a summary like this, it means that you successfully loaded your data.

Training Dataset shape

```
(125, 24)
```

Favorable and unfavorable labels

```
1.0 0.0
```

Protected attribute names

```
['sex']
```

Privileged and unprivileged protected attribute values

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

Dataset feature names

```
['resting_blood_pressure', 'num_major_vessels', 'serum_cholesterol_mg_per_dl', 'oldpeak_eq_st_depression', 'sex', 'ag
e', 'max_heart_rate_achieved', 'slope_of_peak_exercise_st_segment=1', 'slope_of_peak_exercise_st_segment=2', 'slope_o
f_peak_exercise_st_segment=3', 'thal=fixed_defect', 'thal=normal', 'thal=reversible_defect', 'chest_pain_type=1', 'ch
est_pain_type=2', 'chest_pain_type=3', 'chest_pain_type=4', 'fasting_blood_sugar_gt_120_mg_per_dl=0', 'fasting_blood_
sugar_gt_120_mg_per_dl=1', 'resting_ekg_results=0', 'resting_ekg_results=1', 'resting_ekg_results=2', 'exercise_induc
ed_angina=0', 'exercise_induced_angina=1']
```

Step 8:

Location: Application file (manage bias by reweighing.ipynb)
Let’s try our first bias mitigation technique. Copy from or use the “manage bias by reweighing.ipynb” notebook. Change the parts marked by ToDos. Run the whole notebook.

Going further with the AI fairness 360 tool

Now that you know how to load your own data, you can use the example notebooks on the AI Fairness 360 website. The medical expenditure tutorial is a comprehensive example notebook of the usage of many metrics and bias mitigation algorithms.

For more examples, visit the GitHub examples directory.

For more information on the code, visit the GitHub repository.