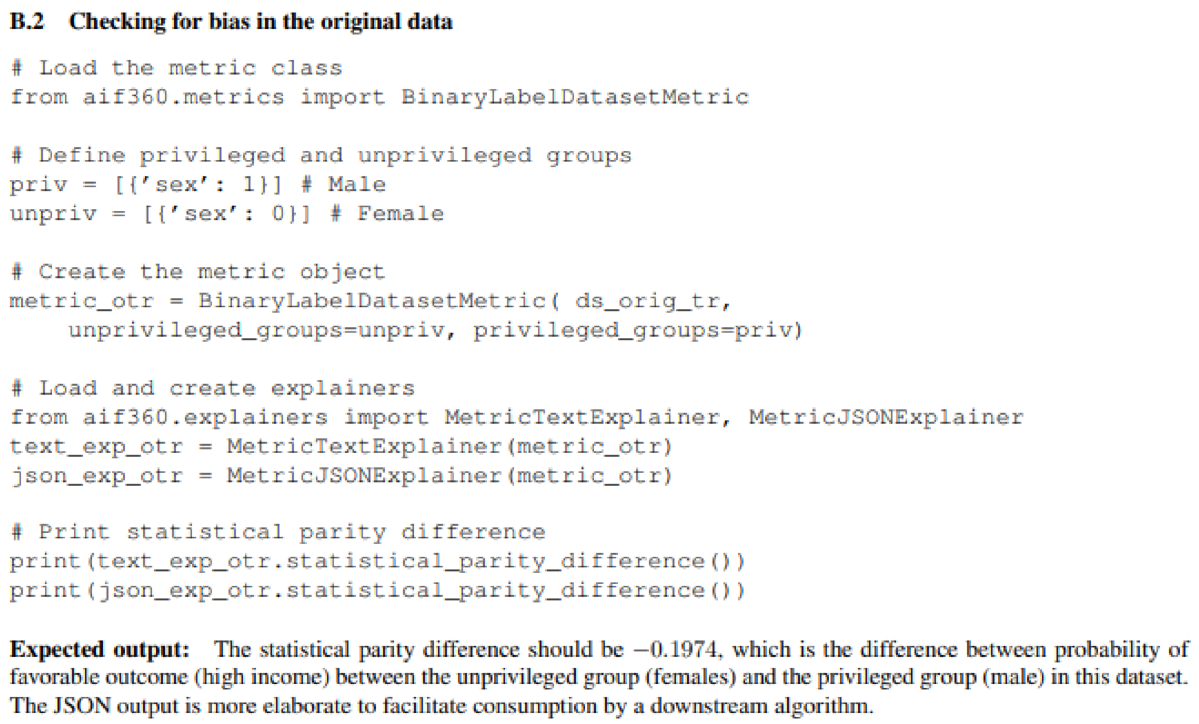
1. Code examples on the AI fairness website https://aif360.mybluemix.net/
    a. AI Fairness Github
2. First work with the Reject option based classification, looks most promising
    a. Watch for a reduction in accuracy
    b. Code to add Reject Option Classification component
    c. Code for Post Processing Reject Option Classification
3. Repeat step 2 with the reweighing
    a. Is this less effective being pre-processing?
    b. Code to test Reweighing component
    c. Code to add Reweighing component
    d. Code for Preprocessing Reweighing
4. Communicate back any problems
    a. Might have to research more possible strategies other than in AI fairness

Link to read more: https://arxiv.org/abs/1810.01943



B.2 Checking for bias in the original data

```
# Load the metric class
from aif360.metrics import BinaryLabelDatasetMetric

# Define privileged and unprivileged groups
priv = [{’sex’: 1}] # Male
unpriv = [{’sex’: 0}] # Female

# Create the metric object
metric_otr = BinaryLabelDatasetMetric( ds_orig_tr,
    unprivileged_groups=unpriv, privileged_groups=priv)

# Load and create explainers
from aif360.explainers import MetricTextExplainer, MetricJSONExplainer
text_exp_otr = MetricTextExplainer(metric_otr)
json_exp_otr = MetricJSONExplainer(metric_otr)

# Print statistical parity difference
print(text_exp_otr.statistical_parity_difference())
print(json_exp_otr.statistical_parity_difference())
```

Expected output: The statistical parity difference should be -0.1974, which is the difference between probability of favorable outcome (high income) between the unprivileged group (females) and the privileged group (male) in this dataset. The JSON output is more elaborate to facilitate consumption by a downstream algorithm.

B.3 Pre-process data to mitigate bias

```
# Import the reweighing preprocessing algorithm class
from aif360.algorithms.preprocessing.reweighing import Reweighing

# Create the algorithm object
RW = Reweighing(unprivileged_groups=unpriv, privileged_groups=priv)

# Train and predict on the training data
```

```
# Uses scikit-learn convention (fit, predict, transform)
RW.fit(ds_orig_tr)
ds_transf_tr = RW.transform(ds_orig_tr)
```

Expected output: There will be no output here, but the reweighing algorithm equalizes the weights across (group, label) combination.

B.4 Checking for bias in the pre-processed training data

```
# Create the metric object for pre-processed data
metric_ttr = BinaryLabelDatasetMetric(ds_transf_tr,
    unprivileged_groups=unpriv, privileged_groups=priv)

# Create explainer
text_exp_ttr = MetricTextExplainer(metric_ttr)

# Print statistical parity difference
print(text_exp_ttr.statistical_parity_difference())
```

Expected output: Because of the action of the re-weighing pre-processing algorithm, the statistical parity difference for the transformed data (`ds_transf_tr`) must be really close to 0.

B.5 Pre-process out-of-sample testing data and check for bias

```
# Apply the learned re-weighing pre-processor
ds_transf_te = RW.transform(ds_orig_te)

# Create metric objects for original and
# pre-processed test data
metric_ote = BinaryLabelDatasetMetric(ds_orig_te,
    unprivileged_groups=unpriv, privileged_groups=priv)
metric_tte = BinaryLabelDatasetMetric(ds_transf_te,
    unprivileged_groups=unpriv, privileged_groups=priv)

# Create explainers for both metric objects
text_exp_ote = MetricTextExplainer(metric_ote)
text_exp_tte = MetricTextExplainer(metric_tte)

# Print statistical parity difference
print(text_exp_ote.statistical_parity_difference())
print(text_exp_tte.statistical_parity_difference())
```

Expected output: The trained re-weighing pre-processor can be applied on the out-of-sample test data. The metrics for the original and transformed testing data will show a significant reduction in statistical parity difference (-0.2021 to -0.0119 in this case).