

## AI Fairness Assignment

To prepare for this assignment, I first updated Anaconda on my local machine to ensure that I had the latest version with all the necessary features and compatibility.

```
(base) C:\Users\vpvai>conda update anaconda
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\vpvai\anaconda3

  added / updated specs:
    - anaconda

The following packages will be downloaded:



| package                             | build          |       |
|-------------------------------------|----------------|-------|
| backports.functools_lru_cache-1.6.4 | pyhd3eb1b0_0   | 9 KB  |
| backports.tempfile-1.0              | pyhd3eb1b0_1   | 11 KB |
| memory_profiler-0.58.0              | pyhd3eb1b0_0   | 31 KB |
| slicer-0.0.7                        | pyhd3eb1b0_0   | 18 KB |
| xmltodict-0.13.0                    | py38haa95532_0 | 19 KB |
| Total:                              |                | 88 KB |



The following packages will be UPDATED:

backports.functools 1.6.1-py_0 --> 1.6.4-pyhd3eb1b0_0
xmltodict           pkgs/main/noarch::xmltodict-0.12.0-py~ --> pkgs/main/win-64::xmltodict-0.13.0-py38haa95532_0
```

After updating, I set up a Python environment within Anaconda and installed the required libraries, including aif360 and fairlearn. Once the setup was complete, I launched a Jupyter Notebook to work through the assignment. I created a dedicated directory to store the project files, including the dataset and notebook.

```
(base) C:\Users\vpvai>conda install -c conda-forge aif360
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\vpvai\anaconda3

  added / updated specs:
    - aif360

The following packages will be downloaded:



| package      | build          |                    |
|--------------|----------------|--------------------|
| conda-4.14.0 | py38haa244fe_0 | 1.0 MB conda-forge |
| Total:       |                | 1.0 MB             |



The following packages will be UPDATED:

conda 4.12.0-py38haa244fe_0 --> 4.14.0-py38haa244fe_0

Proceed ([y]/n)? y

Downloading and Extracting Packages
conda-4.14.0 | 1.0 MB | ##### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

```
(base) C:\Users\vpvai>conda install -c conda-forge fairlearn
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 4.14.0
  latest version: 24.9.2

Please update conda by running

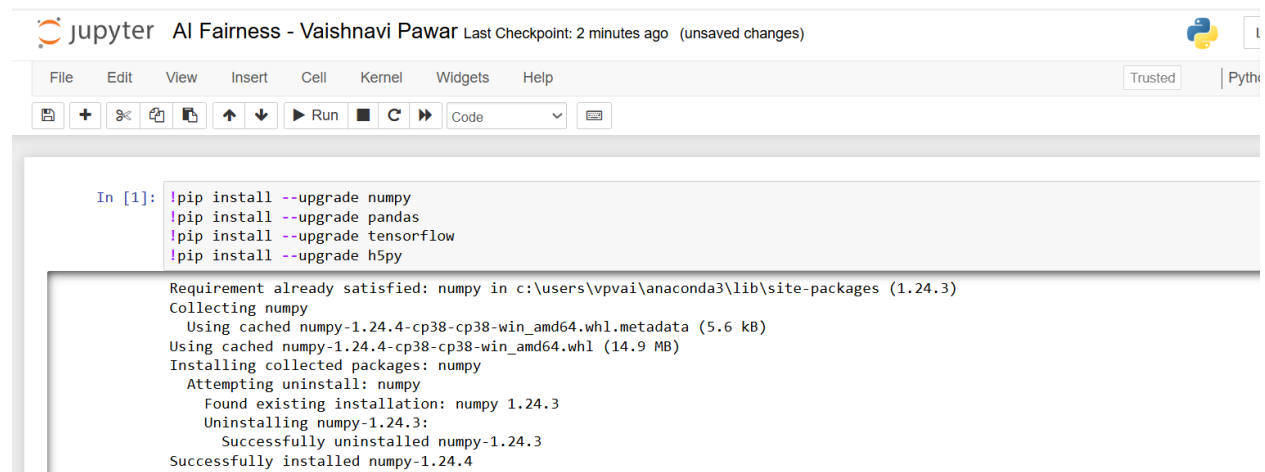
  $ conda update -n base -c conda-forge conda

# All requested packages already installed.

Retrieving notices: ...working... done
```

Following the tutorial, I worked with the AI Fairness 360 toolkit to detect and mitigate bias in the dataset. I verified that all dependencies were properly installed and configured the environment to run smoothly. After configuring the environment, I loaded the dataset into my system with the AI Fairness 360 library's GermanDataset functionality. This enabled me to efficiently load and preprocess the data for bias detection and mitigation.

During the process, I encountered compatibility issues due to outdated versions of key libraries such as NumPy, pandas, TensorFlow, and h5py. These outdated libraries caused errors while running the AI Fairness 360 toolkit. To address this, I upgraded the libraries to their latest versions by executing the following commands:

The screenshot shows a Jupyter Notebook window titled "AI Fairness - Vaishnavi Pawar". The interface includes a top bar with the Jupyter logo, the notebook title, and a "Last Checkpoint: 2 minutes ago (unsaved changes)" status. Below the top bar is a menu bar with options: File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. A toolbar with icons for file operations and execution is also present. The main area displays a code cell with the following commands: 

```
In [1]: !pip install --upgrade numpy
!pip install --upgrade pandas
!pip install --upgrade tensorflow
!pip install --upgrade h5py
```

 The output of the first command is shown below the code cell: 

```
Requirement already satisfied: numpy in c:\users\vpvai\anaconda3\lib\site-packages (1.24.3)
Collecting numpy
  Using cached numpy-1.24.4-cp38-cp38-win_amd64.whl.metadata (5.6 kB)
Using cached numpy-1.24.4-cp38-cp38-win_amd64.whl (14.9 MB)
Installing collected packages: numpy
  Attempting uninstall: numpy
    Found existing installation: numpy 1.24.3
    Uninstalling numpy-1.24.3:
      Successfully uninstalled numpy-1.24.3
  Successfully installed numpy-1.24.4
```

This resolved the compatibility issues and ensured the smooth functioning of the toolkit.

Once the environment was fully operational, I started practicing the steps outlined in the tutorial's example. This helped me understand how to detect and mitigate bias using the AI Fairness 360 toolkit. I closely followed the example, which used the age column as a protected attribute for bias detection and the Reweighting algorithm to mitigate bias. Practicing with this example helped me become acquainted with the toolkit's functionality and workflow, which included loading datasets,

defining privileged and unprivileged groups, calculating fairness metrics, and implementing mitigation techniques. This exercise laid a solid foundation for applying these ideas to my chosen attribute in the following steps.

```
In [3]: import warnings
warnings.filterwarnings('ignore')
import numpy as np
from aif360.datasets import GermanDataset
from aif360.metrics import BinaryLabelDatasetMetric
from aif360.algorithms.preprocessing import Reweighing

In [4]: dataset_orig = GermanDataset(
    protected_attribute_names=['age'],
    privileged_classes=[lambda x: x >= 25],
    features_to_drop=['personal_status', 'sex']
)

In [5]: dataset_orig_train, dataset_orig_test = dataset_orig.split([0.7], shuffle=True)

In [6]: privileged_groups = [{ 'age': 1}]
unprivileged_groups = [{ 'age': 0}]

In [7]: metric_orig_train = BinaryLabelDatasetMetric(dataset_orig_train,
    unprivileged_groups=unprivileged_groups,
    privileged_groups=privileged_groups)

print("Difference in mean outcomes between unprivileged and privileged groups = %f" % metric_orig_train.mean_difference())

Difference in mean outcomes between unprivileged and privileged groups = -0.129093

In [8]: RW = Reweighing(unprivileged_groups=unprivileged_groups,
    privileged_groups=privileged_groups)
dataset_transf_train = RW.fit_transform(dataset_orig_train)

In [9]: metric_transf_train = BinaryLabelDatasetMetric(dataset_transf_train,
    unprivileged_groups=unprivileged_groups,
    privileged_groups=privileged_groups)

print("Difference in mean outcomes between unprivileged and privileged groups = %f" % metric_transf_train.mean_difference())

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

After practicing with the tutorial example, I started working on the assignment. The first step was to explore the dataset and determine which columns were available for analysis. Using the GermanDataset functionality, I retrieved and listed all the dataset's columns to identify potential candidates for the protected attribute. This step was critical because it allowed me to select a column other than the commonly used age or sex, which was required by the assignment criteria. Understanding the dataset structure and attributes enabled me to ensure that my subsequent analysis and bias mitigation efforts were informed and in line with the assignment objectives.

```
In [10]: from aif360.datasets import GermanDataset

# Load the dataset to inspect its columns
dataset = GermanDataset()
print(dataset.feature_names)

['month', 'credit_amount', 'investment_as_income_percentage', 'residence_since', 'age', 'number_of_credits', 'people_liable_for', 'sex', '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']
```

After exploring the dataset and identifying appropriate columns, I chose the **credit\_amount** column as the protected attribute for this assignment. This option allowed me to investigate potential biases based on the amount of credit assigned to individuals. I loaded the dataset with this column set as the protected attribute and prepared it for analysis.

To proceed, I divided the dataset into training and testing sets, with 70% for training and 30% for testing. I then set up privileged and unprivileged groups for the credit\_amount attribute. Privileged groups represented people with more credit, whereas unprivileged groups represented people with less credit. These groups were defined based on specific thresholds to effectively segment the data for bias detection and mitigation. This setup prepared the data for fairness evaluation and mitigation.

```
In [11]: dataset_orig = GermanDataset(
    protected_attribute_names=['credit_amount'],
    privileged_classes=[lambda x: x > 4000],
    features_to_drop=['personal_status', 'age', 'sex']
)
print("Dataset loaded with 'credit_amount' as the protected attribute!")

Dataset loaded with 'credit_amount' as the protected attribute!

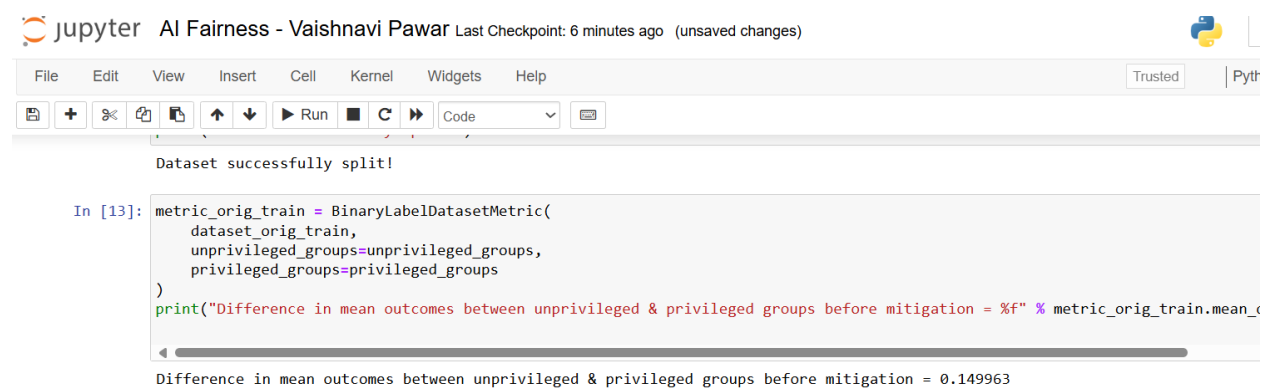
In [12]: # Split the dataset
dataset_orig_train, dataset_orig_test = dataset_orig.split([0.7], shuffle=True)

# Define privileged & unprivileged groups
privileged_groups = [{'credit_amount': 1}]
unprivileged_groups = [{'credit_amount': 0}]

print("Dataset successfully split!")

Dataset successfully split!
```

After preparing the dataset and identifying the groups, I used the AI Fairness 360 toolkit's BinaryLabelDatasetMetric feature to calculate the fairness measures for detecting bias in the training dataset. The **first difference in mean outcomes** between the unprivileged and privileged groups for the **credit\_amount attribute was 0.149963**, indicating a significant divergence in outcomes and the possibility of bias in the dataset.



```
jupyter AI Fairness - Vaishnavi Pawar Last Checkpoint: 6 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted | Pyth

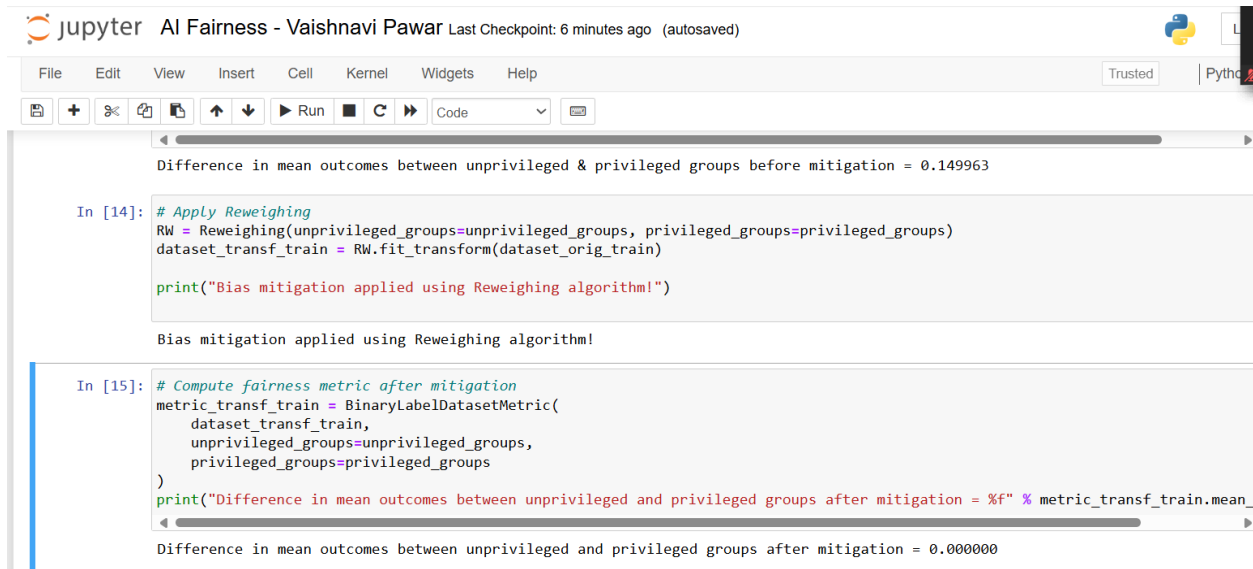
Dataset successfully split!

In [13]: metric_orig_train = BinaryLabelDatasetMetric(
    dataset_orig_train,
    unprivileged_groups=unprivileged_groups,
    privileged_groups=privileged_groups
)
print("Difference in mean outcomes between unprivileged & privileged groups before mitigation = %f" % metric_orig_train.mean_c

Difference in mean outcomes between unprivileged & privileged groups before mitigation = 0.149963
```

Next, I used the AI Fairness 360 toolkit's reweighing method to reduce the discovered bias in the dataset. This algorithm modifies the weights of the training data to guarantee that the protected attribute produces more equitable results for the affluent and underprivileged groups.

I used the Reweighting functionality to alter the training dataset and address the observed inequalities. After implementing this pre-processing bias mitigation strategy, I calculated the fairness metrics again to assess the effectiveness of the mitigation. The **difference in mean outcomes** between the unprivileged and privileged groups after mitigation was **0.000000**, demonstrating that the algorithm was successful in reducing bias to a negligible level.



The screenshot shows a Jupyter Notebook titled "AI Fairness - Vaishnavi Pawar". The interface includes a top bar with the Jupyter logo, the notebook title, and a "Last Checkpoint: 6 minutes ago (autosaved)" status. Below the top bar is a menu bar with options: File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. A toolbar with icons for file operations and execution is also present. The notebook content area displays two code cells. The first cell, labeled "In [14]:", contains code to apply Reweighting and prints a message. The second cell, labeled "In [15]:", contains code to compute a fairness metric and prints the result. The output of the first cell shows a difference of 0.149963, and the output of the second cell shows a difference of 0.000000.

```
Difference in mean outcomes between unprivileged & privileged groups before mitigation = 0.149963

In [14]: # Apply Reweighting
RW = Reweighting(unprivileged_groups=unprivileged_groups, privileged_groups=privileged_groups)
dataset_transf_train = RW.fit_transform(dataset_orig_train)

print("Bias mitigation applied using Reweighting algorithm!")

Bias mitigation applied using Reweighting algorithm!

In [15]: # Compute fairness metric after mitigation
metric_transf_train = BinaryLabelDatasetMetric(
    dataset_transf_train,
    unprivileged_groups=unprivileged_groups,
    privileged_groups=privileged_groups
)
print("Difference in mean outcomes between unprivileged and privileged groups after mitigation = %f" % metric_transf_train.mean_)

Difference in mean outcomes between unprivileged and privileged groups after mitigation = 0.000000
```

## **Report:**

The tutorial was an intriguing trip into recognizing bias in datasets and how to avoid it using the AI Fairness 360 toolbox. This exercise taught me how cultural and structural biases can materialize in machine learning models when specific variables, such as credit amount, age, or gender, are connected with differential results for privileged and underprivileged groups. The toolbox allowed me to properly measure bias by computing the difference in mean outcomes between these groups, and it exposed me to pre-processing mitigation approaches like the Reweighting algorithm. This process stressed the significance of defining fairness indicators, such as demographic parity, to ensure that machine learning models generate equitable results across demographics.

While working through the lesson, I encountered a number of difficulties, including compatibility concerns with obsolete libraries such as NumPy, pandas, TensorFlow, and h5py, which initially hampered the usefulness of the AI Fairness 360 toolset. To resolve these concerns, the libraries must be updated to the most recent versions, ensuring the environment's stability and functionality. Another problem was choosing the right column as the protected attribute to meet the assignment criterion. After reviewing the dataset, I selected the `credit_amount` column, established privileged and unprivileged groups using certain criteria, and assessed fairness metrics. Implementing the Reweighting algorithm successfully reduced bias, as indicated by the difference in mean outcomes lowering to zero, indicating the method's effectiveness.

Overall, this project improved my understanding of fairness in machine learning, emphasizing the need of bias identification and mitigation in real-world applications. It also helped me understand the intricacies of dealing with data and the importance of careful pre-processing to produce ethical and impartial results in predictive models.