# ARTIFICIAL INTELLIGENCE

Detecting y mitigating bias on machine learning models

*Usuario de Microsoft Office*

María Borbonés, AI Solutions Architect
@mariaborbones

**LAB OVERVIEW** ........ **2**

**1. ACCESS TO WATSON STUDIO** ........ **3**

**2. USE CASE** ........ **4**

**3. DATA USED** ........ **4**

**4. AI FAIRNESS 360 GLOSSARY** ........ **5**

**5. ACCESS THE NOTEBOOK** ........ **7**

**6. UNDERSTANDING THE CODE** ........ **9**

5.1 JUPYTER NOTEBOOK ........ 9
5.2 IMPORTING THE PROPER LIBRARIES ........ 10
5.3 DEFINING USEFUL FUNCTIONS ........ 11
5.4 LOAD DATA & SHAPE IT ........ 11
5.5 ANALYZING THE DATA ........ 12
5.6 CHECKING IF OUR DATASET IS BIASED ........ 12
5.7 CREATING A CLASSIFIER USING LOGISTIC REGRESSION ........ 13
5.8 VALIDATING THE MODEL USING THE TEST DATA ........ 13
5.9 TRANSFORMING THE DATA – BIAS MITIGATION ........ 15
5.10 CREATING A CLASSIFIER BASED ON DATA TRANSFORMED BY REWEIGHING ........ 16

**7. ADDITIONAL CONTENT** ........ **17**

6.1 DEPLOYING THE MODEL ........ 17
6.2 GENERATING EXPLANATIONS FOR MODEL PREDICTIONS USING LIME ........ 20

## Lab Overview

On this lab you will learn how to detect and mitigate bias on machine learning models using AI Fairness 360 Open Source toolkit from IBM.

**AI Fairness 360 Open Source Toolkit** is an extensible open source toolkit that can help you examine, report, and mitigate discrimination and bias in machine learning models throughout the AI application lifecycle. Containing over 70 fairness metrics and 10 state-of-the-art bias mitigation algorithms developed by the research community, it is designed to translate algorithmic research from the lab into the actual practice of domains as wide-ranging as finance, human capital management, healthcare, and education. We invite you to use it and improve it. You can get more info on: https://aif360.mybluemix.net/

So, during the lab, we are going to apply the framework on a specific model using **Watson Studio,** the greatest DS & AI platform existing in the market and created by IBM. You can learn more about it on: https://www.ibm.com/cloud/watson-studio. But also, we will show you how enterprises can check and mitigate bias without code using **Watson OpenScale**.

**Requirements:**

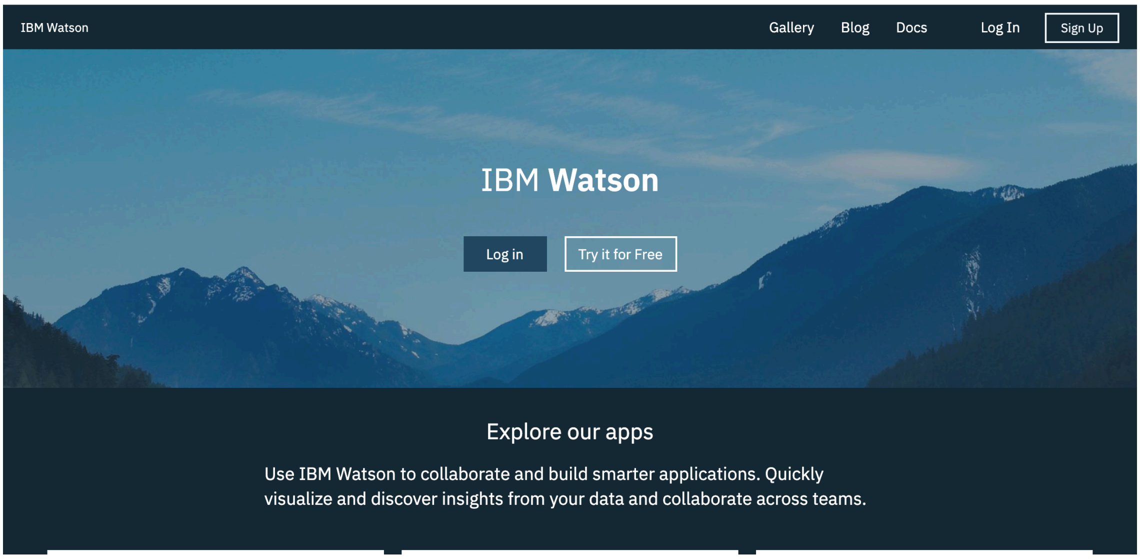
- IBM Cloud account
- Basic Python Knowledge
- Basic Machine Learning Knowledge

**Documentation**

- You can access the code snippets and documentation at: https://github.com/mariaborbones/Codemotion_lab

## 1. Access to Watson Studio

Go to https://eu-de.dataplatform.cloud.ibm.com and log in with the credentials that the instructor has shared with you.

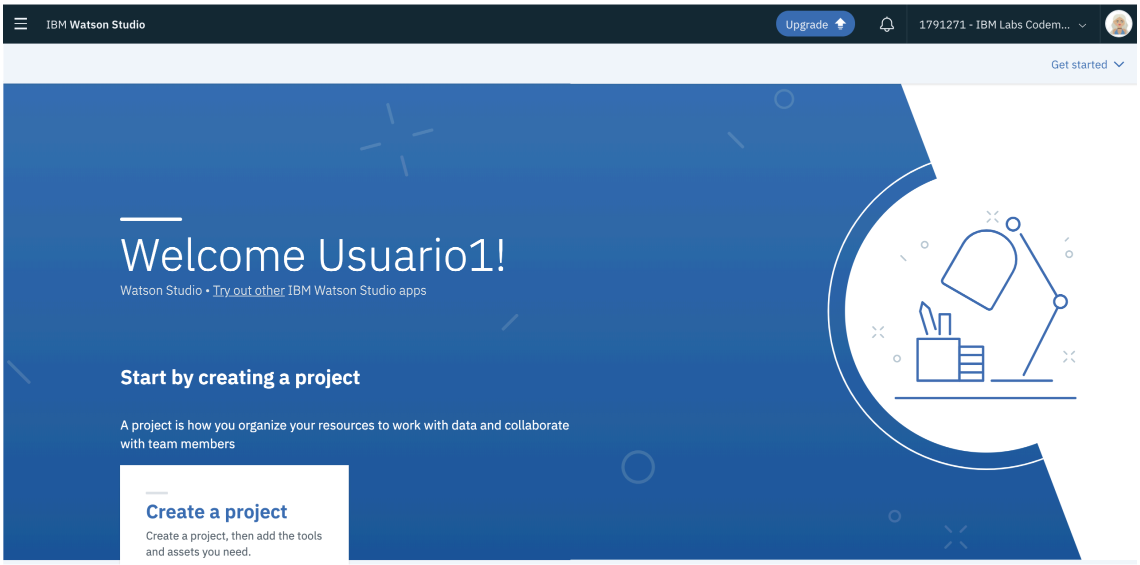


Explore our apps

Use IBM Watson to collaborate and build smarter applications. Quickly visualize and discover insights from your data and collaborate across teams.

Once you are logged in, you can access to the AI project that contains the data and notebooks that we are going to use for today's session.

You will see a welcome message on your screen once you get access to the platform



Welcome Usuario1!

Watson Studio • Try out other IBM Watson Studio apps

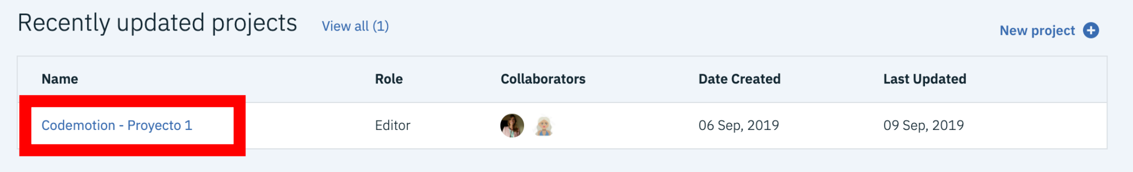
Start by creating a project

A project is how you organize your resources to work with data and collaborate with team members

Create a project

Create a project, then add the tools and assets you need.

Scroll down and you will see the projects where you are part of the team. For this lab, you are going to work on the Codemotion – Proyecto ($user_number) lab.



Recently updated projects View all (1) New project

| Name | Role | Collaborators | Date Created | Last Updated |
|---|---|---|---|---|
| Codemotion - Proyecto 1 | Editor | | 06 Sep, 2019 | 09 Sep, 2019 |

*Well Done! You are now ready to start the lab.*

## 2. Use Case

In order to demonstrate how AIF 360 can be used to detect and mitigate bias in classifier models, we adopt the following use case:

- A data scientist develops a 'fair' healthcare utilization scoring model with respect to defined protected classes. Fairness may be dictated by legal or government regulations, such as a requirement that additional care decisions be not predicated on factors such as race of the patient.
- Developer takes the model AND performance characteristics / specs of the model (e.g. accuracy, fairness tests, etc. basically the model factsheet) and deploys the model in an enterprise app that prioritizes cases for care management.
- The app is put into production and starts scoring people and making recommendations.
- Explanations are generated for each recommendation
- Both recommendations and associated explanations are given to nurses as a part of the care management process. The nurses can evaluate the recommendations for quality and correctness and provide feedback.
- Nurse feedback as well as analysis of usage data with respect to specs of the model w.r.t accuracy and fairness is communicated to AI Ops specialist and LOB user periodically.
- When significant drift in model specs relative to the model factsheet is observed, the model is sent back for retraining.

## 3. Data Used

The specific data used is the 2015 Full Year Consolidated Data File as well as the 2016 Full Year Consolidated Data File.
For each dataset, the sensitive attribute is 'RACE' constructed as follows: 'Whites' (privileged class) defined by the features RACEV2X = 1 (White) and HISPANX = 2 (non Hispanic); 'Non-Whites' that included everyone else.

All the methods we have in the toolkit assume a binary classfier. So we compared fairness where Whites were privileged and non-whites were privileged. Whites are defined as Non-Hispanic Whites (hence Race2vx = 1 and Hispanx = 2). All others were

classified as Non-white. Furthermore,we make the class numeric where privileged/unprivileged are mapped as 1/0.
Along with race as the sensitive feature, other features used for modeling include demographics (such as age, gender, active duty status), physical/mental health assessments, diagnosis codes (such as history of diagnosis of cancer, or diabetes), and limitations (such as cognitive or hearing or vision limitation).
To measure utilization, a composite feature, 'UTILIZATION', was created to measure the total number of trips requiring some sort of medical care by summing up the following features: OBTOTV15(16), the number of office based visits; OPTOTV15(16), the number of outpatient visits; ERTOT15(16), the number of ER visits; IPNGTD15(16), the number of inpatient nights, and + HHTOTD16, the number of home health visits.
The model classification task is to predict whether a person would have 'high' utilization (defined as UTILIZATION >= 10, roughly the average utilization for the considered population). High utilization respondents constituted around 17% of each dataset.
To simulate the scenario, each dataset is split into 3 parts: a train, a validation, and a test/deployment part.
We assume that the model is initially built and tuned using the 2015 Panel 19 train/test data. It is then put into practice and used to score people to identify potential candidates for care management. Initial deployment is simulated to 2015 Panel 20 deployment data. To show change in performance and/or fairness over time, (use case steps 6-7), the 2016 Panel 21 deployment data is used. Finally, if drift is observed, the 2015 train/validation data is used to learn a new model and evaluated again on the 2016 deployment data

## 4. AI Fairness 360 Glossary

**Bias**
A systematic error. In the context of fairness, we are concerned with unwanted bias that places privileged groups at systematic advantage and unprivileged groups at systematic disadvantage.

**Bias mitigation algorithm**
A procedure for reducing unwanted bias in training data or models.

**Classifier**
A model that predicts categorical labels from features.

**Explainer**
Functionality for providing details on or causes for fairness metric results.

**Fairness metric**
A quantification of unwanted bias in training data or models.

**Favorable label**
A label whose value corresponds to an outcome that provides an advantage to the recipient. The opposite is an unfavorable lable.

**Feature**
An attribute containing information for predicting the label.

**Group fairness**
The goal of groups defined by protected attributes receiving similar treatments or outcomes.

**In-processing algorithm**
A bias mitigation algorithm that is applied to a model during its training.

**Individual fairness**
The goal of similar individuals receiving similar treatments or outcomes.

**Instance weight**
A numerical value that multiplies the contribution of a data point in a model.

**Label**
A value corresponding to an outcome.

**Machine learning**
A general approach for determining models from data.

**Model**
A function that takes features as input and predicts labels as output.

**Post-processing algorithm**
A bias mitigation algorithm that is applied to predicted labels.

**Pre-processing algorithm**
A bias mitigation algorithm that is applied to training data.

**Privileged protected attribute**
A value of a protected attribute indicating a group that has historically been at systematic advantage.

**Protected attribute**
An attribute that partitions a population into groups whose outcomes should have parity. Examples include race, gender, caste, and religion. Protected attributes are not universal, but are application specific.

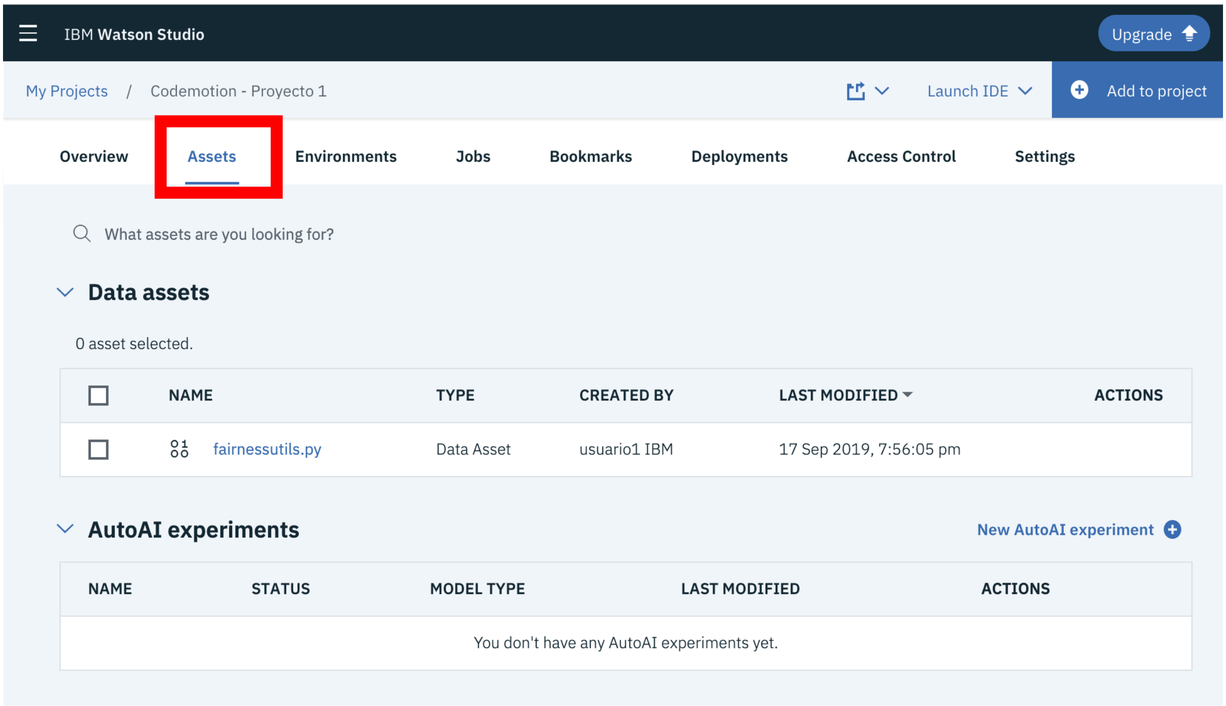
**Score**
A continuous value output from a classifier. Applying a threshold to a score results in a predicted label.

**Training data**
A dataset from which a model is learned.

**Transformer**
A procedure that modifies a dataset.

## 5. Access the notebook

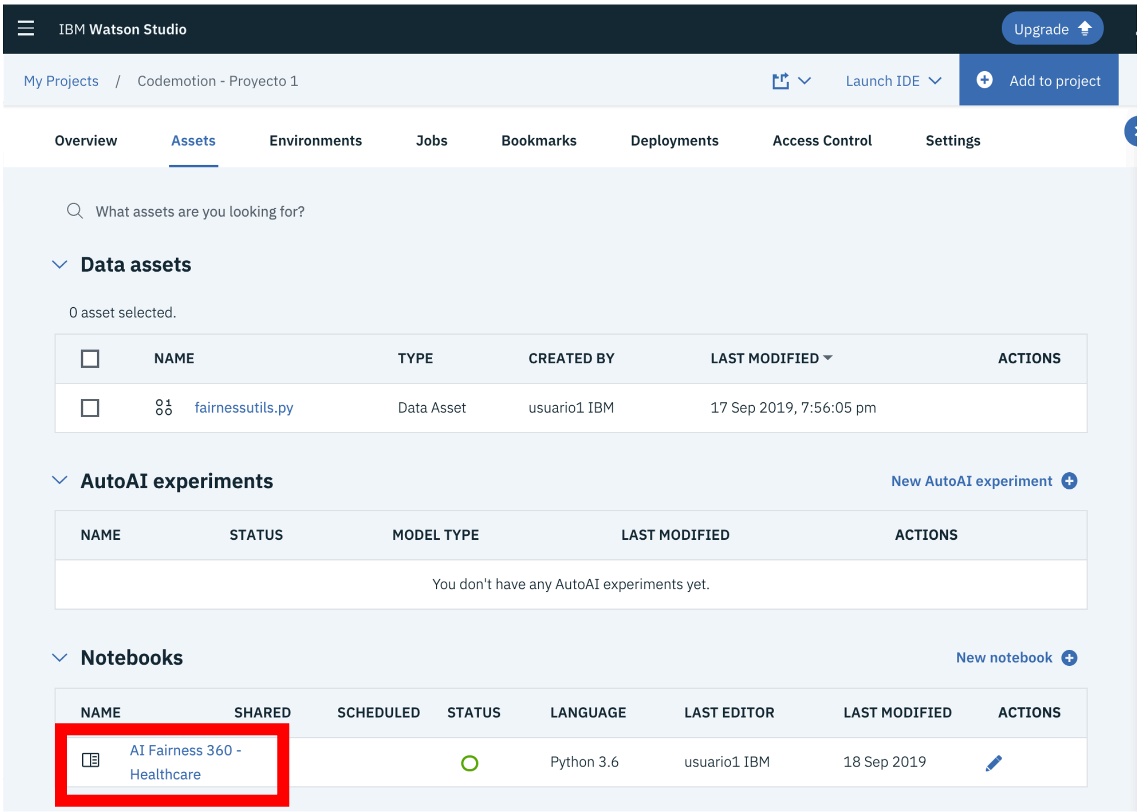
Go to your project and access the **assets** tab:



| □ | NAME | TYPE | CREATED BY | LAST MODIFIED | ACTIONS |
|---|---|---|---|---|---|
| □ | fairnessutils.py | Data Asset | usuario1 IBM | 17 Sep 2019, 7:56:05 pm | |

| NAME | STATUS | MODEL TYPE | LAST MODIFIED | ACTIONS |
|---|---|---|---|---|
| You don't have any AutoAI experiments yet. | | | | |

You will find a notebook, under notebooks section, named as **AI Fairness 360 – Healthcare**



**Data assets**

0 asset selected.

| □ | NAME | TYPE | CREATED BY | LAST MODIFIED | ACTIONS |
|---|---|---|---|---|---|
| □ | fairnessutils.py | Data Asset | usuario1 IBM | 17 Sep 2019, 7:56:05 pm | |

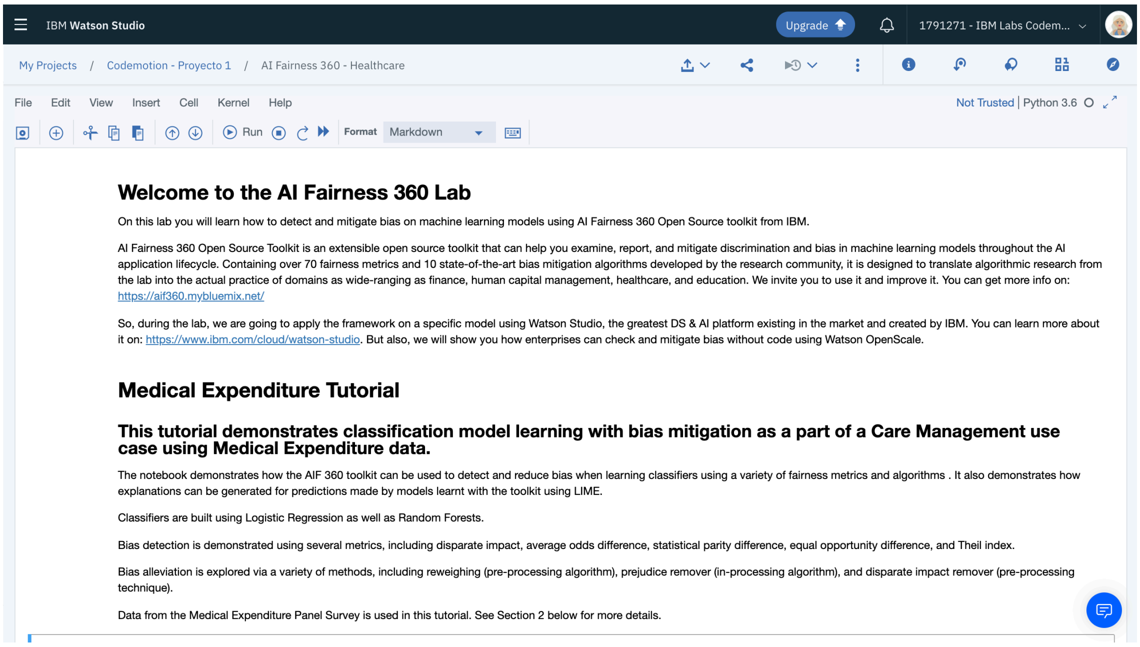
**AutoAI experiments**

| NAME | STATUS | MODEL TYPE | LAST MODIFIED | ACTIONS |
|---|---|---|---|---|
| You don't have any AutoAI experiments yet. | | | | |

**Notebooks**

| NAME | SHARED | SCHEDULED | STATUS | LANGUAGE | LAST EDITOR | LAST MODIFIED | ACTIONS |
|---|---|---|---|---|---|---|---|
| AI Fairness 360 - Healthcare | | | ○ | Python 3.6 | usuario1 IBM | 18 Sep 2019 | |

Click on it to access its content


IBM Watson Studio
Upgrade
1791271 - IBM Labs Codem...
My Projects / Codemotion - Proyecto 1 / AI Fairness 360 - Healthcare
File
Edit
View
Insert
Cell
Kernel
Help
Not Trusted
Python 3.6
Run
Format
Markdown

**Welcome to the AI Fairness 360 Lab**

On this lab you will learn how to detect and mitigate bias on machine learning models using AI Fairness 360 Open Source toolkit from IBM.

AI Fairness 360 Open Source Toolkit is an extensible open source toolkit that can help you examine, report, and mitigate discrimination and bias in machine learning models throughout the AI application lifecycle. Containing over 70 fairness metrics and 10 state-of-the-art bias mitigation algorithms developed by the research community, it is designed to translate algorithmic research from the lab into the actual practice of domains as wide-ranging as finance, human capital management, healthcare, and education. We invite you to use it and improve it. You can get more info on: https://aif360.mybluemix.net/

So, during the lab, we are going to apply the framework on a specific model using Watson Studio, the greatest DS & AI platform existing in the market and created by IBM. You can learn more about it on: https://www.ibm.com/cloud/watson-studio. But also, we will show you how enterprises can check and mitigate bias without code using Watson OpenScale.

**Medical Expenditure Tutorial**

**This tutorial demonstrates classification model learning with bias mitigation as a part of a Care Management use case using Medical Expenditure data.**

The notebook demonstrates how the AIF 360 toolkit can be used to detect and reduce bias when learning classifiers using a variety of fairness metrics and algorithms . It also demonstrates how explanations can be generated for predictions made by models learnt with the toolkit using LIME.

Classifiers are built using Logistic Regression as well as Random Forests.

Bias detection is demonstrated using several metrics, including disparate impact, average odds difference, statistical parity difference, equal opportunity difference, and Theil index.

Bias alleviation is explored via a variety of methods, including reweighing (pre-processing algorithm), prejudice remover (in-processing algorithm), and disparate impact remover (pre-processing technique).

Data from the Medical Expenditure Panel Survey is used in this tutorial. See Section 2 below for more details.

*Congrats! Now you can start executing and understanding the code*

## 6. Understanding the code

The notebook demonstrates how the AIF 360 toolkit can be used to detect and reduce bias when learning classifiers using a variety of fairness metrics and algorithms. It also demonstrates how explanations can be generated for predictions made by models learnt with the toolkit using LIME.

Classifiers are built using Logistic Regression and we will add Random Forest.

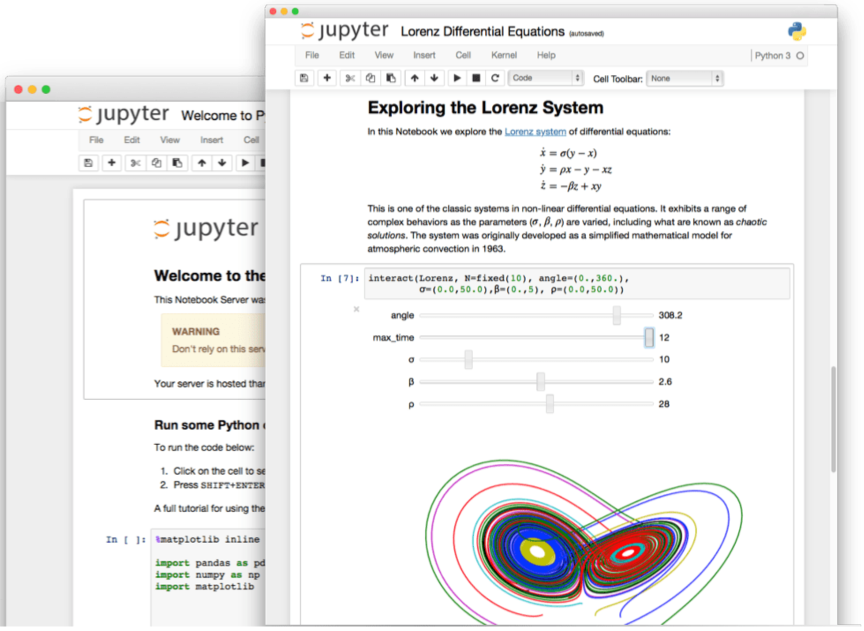
Bias detection is demonstrated using several metrics, including disparate impact, average odds difference, statistical parity difference, equal opportunity difference, and Theil index.

Bias alleviation is explored via a variety of methods, including reweighing (pre-processing algorithm), prejudice remover (in-processing algorithm), and disparate impact remover (pre-processing technique).

### 5.1 Jupyter Notebook

Watson Studio Notebooks are based on Jupyter Notebooks. If it is your first time workin in a Jupyter Notebook, let me introduce them to you. If not, you can skip this section.

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.



Welcome to the

This Notebook Server wa

WARNING

Don't rely on this serv

Your server is hosted than

Run some Python

To run the code below:

1. Click on the cell to se
2. Press SHIFT+ENTER

A full tutorial for using the

```
In [ ]: %matplotlib inline

        import pandas as pd
        import numpy as np
        import matplotlib
```

Exploring the Lorenz System

In this Notebook we explore the Lorenz system of differential equations:

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = \rho x - y - xz$$
$$\dot{z} = -\beta z + xy$$

This is one of the classic systems in non-linear differential equations. It exhibits a range of complex behaviors as the parameters ($\sigma$, $\beta$, $\rho$) are varied, including what are known as *chaotic solutions*. The system was originally developed as a simplified mathematical model for atmospheric convection in 1963.

```
In [7]: interact(Lorenz, N=fixed(10), angle=(0.,360.),
                 σ=(0.0,50.0),β=(0.,5), ρ=(0.0,50.0))
```

angle 308.2

max_time 12

σ 10

β 2.6

ρ 28

IBM Watson Studio leverages Jupyter Notebooks with additional capabilities, such as, social sharing, backups, collaborative tolos and data access.

### 5.2 Importing the proper libraries

In order to be able to work with the AI Fairness 360 toolkit, we will need to install the **aif360** library that is available trough pip install.

The notebook you are working on, already contains the library but for future projects you will need to install it as follows:

```
!pip install aif360
```

You will notice that the first step on the notebook is to import the necessary packages.

**Cell Number: #1**
**Action to be taken:** Select cell number 1 and click RUN

The main packages are:

**Aif360 datasets:** They contain the data available for testing the framework. IBM researchers have created some methods to access and manipulate the data. You can get more information here: https://aif360.readthedocs.io/en/latest/modules/datasets.html

**Aif360 metrics:** they import the methods to detect bias on data, algorithms and predictions. More info: https://aif360.readthedocs.io/en/latest/modules/metrics.html

**Aif360 algorithms:** they import the algorithms to mitigate bias on data, algorithms and predictions. More info: https://aif360.readthedocs.io/en/latest/modules/algorithms.html

**Aif360 explainer:** they import explainers to explain on natural language the meaning of AIf360 metrics. More info:
https://aif360.readthedocs.io/en/latest/modules/explainers.html

Today, there are so many frameworks available to work on machine learning models. For this lab, we have chosen to use scikit learn https://scikit-learn.org/stable/. It is a framework built on Numpy, Scipy and matplotlib for data mining and data analysis. That's why you will find imports from the sklearn library, such as, **sklearn.preprocessing**

Additionally, we will use at the end of the lab the **Lime** project. Lime is about explaining what machine classifiers are doing. At the moment, they support explaining individual predictions for text classifiers or classifiers that act on tables. You can find more information about it on this paper: https://arxiv.org/abs/1602.04938

### 5.3 Defining useful functions

Next step is to define all the needed functions for our lab.

**Cell Number: #2**
**Action to be taken:** Select cell number 2 and click RUN

We are going to define mainly four methods:

- **def test(dataset, model, thresh_arr):** This function is used to test the generated model over dataset and get the associated metrics

- **def describe_metrics(metrics, thresh_arr):** Displays associated fairness metrics
- **def describe(train=None, val=None, test=None):** Describes the dataset characteristics

### 5.4 Load Data & shape it

Now it is time to load the data. On section 2 of the notebook "**Load data & create splits for learning/validating/testing model",** we are going to load the dataset and Split i tinto train, validate and test datasets. This will allow us to train and test our model to find the best version of it.

The method **MEPSDataset19()** invokes a aif360 method that brings the data to stage.

```
(dataset_orig_panel19_train,
 dataset_orig_panel19_val,
 dataset_orig_panel19_test) = MEPSDataset19().split([0.5, 0.8], shu
ffle=True)
```

The above code creates three different datasets as I already mentioned. And then, sets the feature RACE as the protected attribute and sets its value as:

```
[{'RACE': 1.0}] : privileged_groups
[{'RACE': 0.0}] : unprivileged_groups
```

**Cell Number: #3**
**Action to be taken:** Select cell number 3 and click RUN

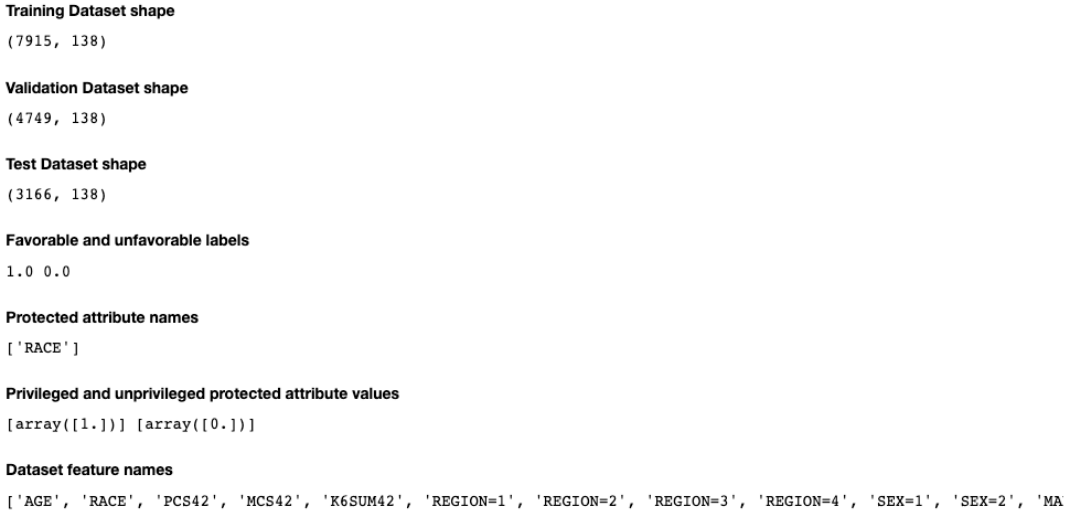
### 5.5 Analyzing the data

Now, we are going to get the 2015 dataset details. For that purpose we use the describe function whose behavior has been already defined on section 5.3 Defining useful functions

We send the three different generated datasets to the function (train,val and test) to get their descriptions.

**Cell Number: #4**
**Action to be taken:** Select cell number 4 and click RUN

You should get something similar to this as a result:



```
Training Dataset shape

(7915, 138)

Validation Dataset shape

(4749, 138)

Test Dataset shape

(3166, 138)

Favorable and unfavorable labels

1.0 0.0

Protected attribute names

['RACE']

Privileged and unprivileged protected attribute values

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

Dataset feature names

['AGE', 'RACE', 'PCS42', 'MCS42', 'K6SUM42', 'REGION=1', 'REGION=2', 'REGION=3', 'REGION=4', 'SEX=1', 'SEX=2', 'MA
```

You can check that our favorable label is 1.0 and unfavorable is 0.0, that the protected attribute has been set as 'RACE' and the datasets shape and feature names are also being displayed.

### 5.6 Checking if our dataset is biased

It's time to apply the **AI fairness 360 toolkit.** Let's check if our dataset is biased or note. In this case we are going to use the Disparate impact metric.

We have already mentioned that the sensitive attribute is 'RACE' constructed as follows: 'Whites' (privileged class) with value 1.0 and HISPANX and 'Non-Whites' as 0.0

The Disparate Impact Metric measures is computed as the ratio of rate of favorable outcome for the unprivileged group to that of the privileged group.

The ideal value of this metric is 1.0 A value < 1 implies higher benefit for the privileged group and a value >1 implies a higher benefit for the unprivileged group.

Fairness for this metric is between 0.8 and 1.2

**Cell Number: #5**
**Action to be taken:** Select cell number 5 and click RUN

As result you should get something similar to:

```
Disparate impact (probability of favorable outcome for unprivileged
instances / probability of favorable outcome for privileged instanc
es): 0.48230522996275893
```

The result of the disparate impact means that there is more probability of getting a favorable outcome for the privileged group. **So Data is biased**

**Additional Info**:
MetricTextExplainer(): Class for explaining metric values with text
BinaryLabelDatasetMetric(): Class for computing metrics based on a single BinaryLabelDataset.

### 5.7 Creating a classifier using Logistic Regression

Now we are going to use scikit learn (I already explained what is to you) to build our classifier to predict the level of utilization for our patients or in other words, their medical expenditure.

On section 3.1 of the notebook you can find the piece of code corresponding for the training session. First, we are going to create a pipeline used to help automate **machine learning** workflows and put together different pieces of the Machine Learning process.

```
model = make_pipeline(StandardScaler(),
                      LogisticRegression(solver='liblinear', random
_state=1))
```

On the pipeline, first we call the **standard scaler** function that Standardize features by removing the mean and scaling to unit variance. Standardization of a dataset is a common requirement for many machine learning estimators: they might behave badly if the individual features do not more or less look like standard normally distributed data.

Then, we call the **LogisticRegression** method, that creates a new logistic regression classifier.

Using the **model.fit** method we start executing the pipeline over the training data.

**Cell Number: #6**
**Action to be taken:** Select cell number 6 and click RUN

### 5.8 Validating the model using the test data

Now we want to test our model against the test data. In order to do that, we are going to use the already defined **test function**.

Go to section 1 to check the test method code and all the actions that take place when calling the method.

Within this function we run our classification model over the testing data:

```
model.predict_proba(dataset.features)
```

On section 3.2 from the notebook, you will see that we have defined a threshold array using the **np.linspace(0.01,0.5,50)** function. This function creates a 50-element array starting on 0.01 value ending up on 0.5 value

What is the role of the threshold array? Well, Logistic regression returns a probability. You can return the probability as is or return the probability as a binary value. That's why we use the threshold array and generate a new dataset with the classification result.

Once we get the data needed to understand the validation process result, we use the

```
ClassificationMetric(
                dataset, dataset_pred,
                unprivileged_groups=unprivileged_groups,
                privileged_groups=privileged_groups)
```

method, that give us information about the model accuracy and tests several fairness metrics.

**Cell Number: #7,8,9**
**Action to be taken:** Select cell numbers 7, 8, 9 and click RUN

You should get a similar result:

```
Threshold corresponding to Best balanced accuracy: 0.1900
Best balanced accuracy: 0.7759
Corresponding 1-min(DI, 1/DI) value: 0.5738
Corresponding average odds difference value: -0.2057
Corresponding statistical parity difference value: -0.2612
Corresponding equal opportunity difference value: -0.2228
Corresponding Theil index value: 0.0921
```

For all the fairness metrics displayed above, the value should be close to '0' for fairness.

1-min(DI, 1/DI) < 0.2 is typically desired for classifier predictions to be fair. However, for a logistic regression classifier trained with original training data, at the best classification rate, this is quite high. This implies unfairness.

Similarly, average odds difference=(FPRunpriv−FPRpriv)+(TPRunpriv−TPRpriv)2 must be close to zero for the classifier to be fair.

Again, the results for this classifier-data combination are still high. This still implies unfairness.

Some guidance about the different metrics can be found here:

**Average Odds Difference:** Computed as average difference of false positive rate (false positives / negatives) and true positive rate (true positives / positives) between unprivileged and privileged groups.

The ideal value of this metric is 0. A value of < 0 implies higher benefit for the privileged group and a value > 0 implies higher benefit for the unprivileged group.

Fairness for this metric is between -0.1 and 0.1

**Statistical Parity Difference:** Computed as the difference of the rate of favorable outcomes received by the unprivileged group to the privileged group.

The ideal value of this metric is 0

Fairness for this metric is between -0.1 and 0.1

**Equal Opportunity Difference:** This metric is computed as the difference of true positive rates between the unprivileged and the privileged groups. The true positive rate is the ratio of true positives to the total number of actual positives for a given group.

The ideal value is 0. A value of < 0 implies higher benefit for the privileged group and a value > 0 implies higher benefit for the unprivileged group.

Fairness for this metric is between -0.1 and 0.1

**Theil Index:** Computed as the generalized entropy of benefit for all individuals in the dataset, with alpha = 1. It measures the inequality in benefit allocation for individuals.

A value of 0 implies perfect fairness.

Fairness is indicated by lower scores, higher scores are problematic

## 5.9 Transforming the data – Bias mitigation

So, we already have created our classifier, but we did it on biased data. Now, we want to use a pre-processing technique, specifically reweighing to mitigate bias on our dataset.

On the section 4.1 of our dataset, we are going to apply the reweighing technique. Reweighing is a preprocessing technique that Weights the examples in each (group, label) combination differently to ensure fairness before classification.

Then we apply again our fairness metric, specifically the Disparate Impact, to detect if our dataset is still biased.

**Cell Number: #10,11**
**Action to be taken:** Select cell numbers 10,11 and click RUN

You should get something similar to this:

```
Disparate impact (probability of favorable outcome for unprivileged
instances / probability of favorable outcome for privileged instanc
es): 1.0000000000000002
```

This means that now our dataset is enough fair since we have already mentioned that on the disparate impact metric fairness is between 0.8 and 1.2.

### 5.10 Creating a classifier based on data transformed by reweighing

Now, we are going to repeat steps 5.7 and 5.8 but this time over the resulting transformed data.

This time I'm going to give you the code and you need to insert it on the cells.

First, create a cell to train the model and add this code: (be careful with spaces and tabs)

```
dataset = dataset_transf_panel19_train
model = make_pipeline(StandardScaler(),
                      LogisticRegression(solver='liblinear',
random_state=1))
fit_params = {'logisticregression__sample_weight':
dataset.instance_weights}
lr_transf_panel19 = model.fit(dataset.features,
dataset.labels.ravel(), **fit_params)
```

**Cell Number: #12**
**Action to be taken:** Insert code. Select cell number 12 and click RUN

Then, we need to test our model as we did on previous sections. So, create a new code cell and add the following code:

```
thresh_arr = np.linspace(0.01, 0.5, 50)
val_metrics = test(dataset=dataset_orig_panel19_val,
                   model=lr_transf_panel19,
                   thresh_arr=thresh_arr)
lr_transf_best_ind = np.argmax(val_metrics['bal_acc'])
lr_transf_metrics = test(dataset=dataset_orig_panel19_test,
                         model=lr_transf_panel19,

thresh_arr=[thresh_arr[lr_transf_best_ind]])
```

**Cell Number: #13**
**Action to be taken:** Insert code. Select cell number 13 and click RUN

Finally, add a last cell to get the associated metrics:

```
describe_metrics(lr_transf_metrics,
[thresh_arr[lr_transf_best_ind]])
```

**Cell Number: #14**
**Action to be taken:** Insert code. Select cell number 13 and click RUN

You should get a similar result to this one:

```
Threshold corresponding to Best balanced accuracy: 0.2200
Best balanced accuracy: 0.7539
Corresponding 1-min(DI, 1/DI) value: 0.2482
Corresponding average odds difference value: -0.0151
Corresponding statistical parity difference value: -0.0872
Corresponding equal opportunity difference value: -0.0035
Corresponding Theil index value: 0.0966
```

As a result of applying the reweighing tecnique, the fairness metrics for the logistic regression model are well improved, and thus the model is much fairer relative to the logistic regression model learned from the original data.

*Congrats!!! You have detected and mitigated bias on a machine learning model using AI Fairness 360.*

## 7. Additional content

### 6.1 Deploying the model

On this section of the tutorial we are going to apply the deployed model trained on 2014 data over data from 2015.

So, as you can check on section 5.1, the first thing we are going to do is to generate the dataset from Panel 20 (2015):

```
dataset_orig_panel20_deploy = MEPSDataset20()
```

And then we use the already preconfigured align_dataset method over the new dataset to get the same structure on both datasets.

```
dataset_orig_panel20_deploy =
dataset_orig_panel19_train.align_datasets(dataset_orig_panel20_depl
oy)
```

As we did with the previous dataset, we can use the describe function.

**Cell Number: #15,16**
**Action to be taken:** Select cell numbers 15,16 and click RUN

We will get the following result (quite similar to the previous one)

**Test Dataset shape**

```
(17570, 138)
```

**Favorable and unfavorable labels**

```
1.0 0.0
```

**Protected attribute names**

```
['RACE']
```

**Privileged and unprivileged protected attribute values**

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

**Dataset feature names**

```
['AGE', 'RACE', 'PCS42', 'MCS42', 'K6SUM42', 'REGION=1', 'REGION=2', '
REGION=3', 'REGION=4', 'SEX=1', 'SEX=2', 'MARRY=1', 'MARRY=2', 'MARRY=
3', 'MARRY=4', 'MARRY=5', 'MARRY=6', 'MARRY=7', 'MARRY=8', 'MARRY=9',
'MARRY=10', 'FTSTU=-1', 'FTSTU=1', 'FTSTU=2', 'FTSTU=3', 'ACTDTY=1', '
ACTDTY=2', 'ACTDTY=3', 'ACTDTY=4', 'HONRDC=1', 'HONRDC=2', 'HONRDC=3',
'HONRDC=4', 'RTHLTH=-1', 'RTHLTH=1', 'RTHLTH=2', 'RTHLTH=3', 'RTHLTH=4
', 'RTHLTH=5', 'MNHLTH=-1', 'MNHLTH=1', 'MNHLTH=2', 'MNHLTH=3', 'MNHLT
H=4', 'MNHLTH=5', 'HIBPDX=-1', 'HIBPDX=1', 'HIBPDX=2', 'CHDDX=-1', 'CH
DDX=1', 'CHDDX=2', 'ANGIDX=-1', 'ANGIDX=1', 'ANGIDX=2', 'MIDX=-1', 'MI
DX=1', 'MIDX=2', 'OHRTDX=-1', 'OHRTDX=1', 'OHRTDX=2', 'STRKDX=-1', 'ST
RKDX=1', 'STRKDX=2', 'EMPHDX=-1', 'EMPHDX=1', 'EMPHDX=2', 'CHBRON=-1',
'CHBRON=1', 'CHBRON=2', 'CHOLDX=-1', 'CHOLDX=1', 'CHOLDX=2', 'CANCERDX
=-1', 'CANCERDX=1', 'CANCERDX=2', 'DIABDX=-1', 'DIABDX=1', 'DIABDX=2',
'JTPAIN=-1', 'JTPAIN=1', 'JTPAIN=2', 'ARTHDX=-1', 'ARTHDX=1', 'ARTHDX=
2', 'ARTHTYPE=-1', 'ARTHTYPE=1', 'ARTHTYPE=2', 'ARTHTYPE=3', 'ASTHDX=1
', 'ASTHDX=2', 'ADHDADDX=-1', 'ADHDADDX=1', 'ADHDADDX=2', 'PREGNT=-1',
'PREGNT=1', 'PREGNT=2', 'WLKLIM=-1', 'WLKLIM=1', 'WLKLIM=2', 'ACTLIM=-
1', 'ACTLIM=1', 'ACTLIM=2', 'SOCLIM=-1', 'SOCLIM=1', 'SOCLIM=2', 'COGL
IM=-1', 'COGLIM=1', 'COGLIM=2', 'DFHEAR42=-1', 'DFHEAR42=1', 'DFHEAR42
=2', 'DFSEE42=-1', 'DFSEE42=1', 'DFSEE42=2', 'ADSMOK42=-1', 'ADSMOK42=
1', 'ADSMOK42=2', 'PHQ242=-1', 'PHQ242=0', 'PHQ242=1', 'PHQ242=2', 'PH
Q242=3', 'PHQ242=4', 'PHQ242=5', 'PHQ242=6', 'EMPST=-1', 'EMPST=1', 'E
MPST=2', 'EMPST=3', 'EMPST=4', 'POVCAT=1', 'POVCAT=2', 'POVCAT=3', 'PO
VCAT=4', 'POVCAT=5', 'INSCOV=1', 'INSCOV=2', 'INSCOV=3']
```

Then we check again using disparate impact if the model is biased. In order to accomplish that, add this code to the **Cell 17:**

```
metric_orig_panel20_deploy = BinaryLabelDatasetMetric(
        dataset_orig_panel20_deploy,
        unprivileged_groups=unprivileged_groups,
        privileged_groups=privileged_groups)
explainer_orig_panel20_deploy =
MetricTextExplainer(metric_orig_panel20_deploy)
```

```
print(explainer_orig_panel20_deploy.disparate_impact())
```

You should get the following result:

```
Disparate impact (probability of favorable outcome for unprivileged
instances / probability of favorable outcome for privileged instanc
es): 0.5456992351196291
```

That means that the dataset is also biased.

Test now the model using the 2015 data adding the following code on **Cell 18**:

```
lr_transf_metrics_panel20_deploy = test(
        dataset=dataset_orig_panel20_deploy,
        model=lr_transf_panel19,
        thresh_arr=[thresh_arr[lr_transf_best_ind]])
```

And get the corresponding metrics adding the following code into **Cell 19:**

```
describe_metrics(lr_transf_metrics_panel20_deploy,
[thresh_arr[lr_transf_best_ind]])
```

### 6.2 Generating Explanations for model predictions using LIME

We already explained you what is LIME (go to definition). Now we are going to use LIME to get explanations on 2015 Panel 20 deployment data.

Explanations is one of the main challenges that we are facing today on AI. Some algorithms act as blackboxes and is difficult to get information about why the took a certain decisión or predicted a specific value.

On this section of the lab, we review how LIME can be integrated with AIF360 to get explanations for model predictions.

On the section 6.1 of the notebook you can see how this code transforms aif360 features to LIME-compatible features:

```
s_train = lime_data.transform(train_dataset.features)
s_test = lime_data.transform(test_dataset.features)
```

Then, we create a Tabular explainer. This explainer ompute statistics on each feature (column). If the feature is numerical, it computes the mean and std, and discretize it into quartiles. If the feature is categorical, it computes the frequency of each value.

We use these computed statistics for two things:

1. To scale the data, so that we can meaningfully compute distances when the attributes are not on the same scale
2. To sample perturbed instances - which we do by sampling from a Normal(0,1), multiplying by the std and adding back the mean.

Then we create two methods:

The **inverse_transform()** function is used to transform LIME-compatible data back to aif360-compatible data since that is needed by the model to make predictions. The function below is used to produce the predictions for any perturbed data that is produce by LIME

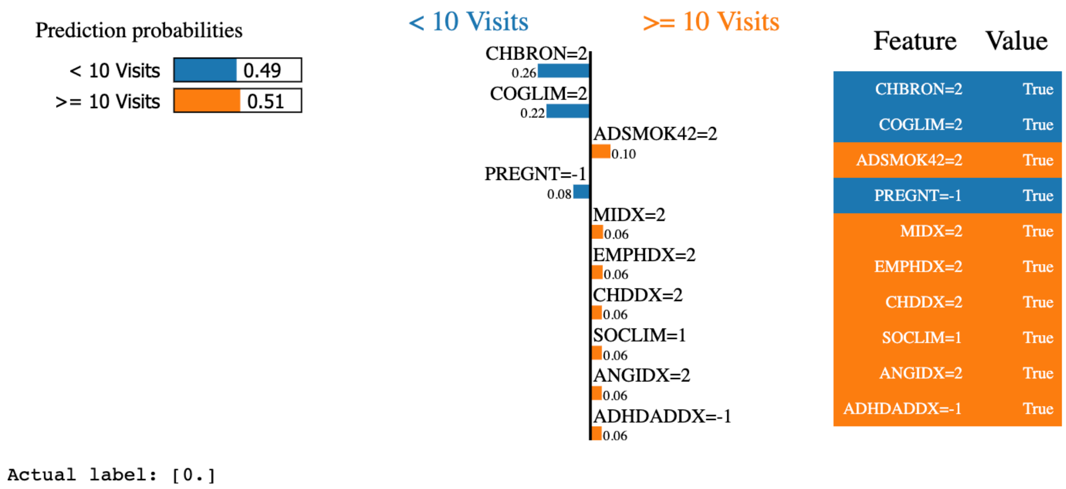
The **explain_instance()** method can then be used to produce explanations for any instance in the test dataset

Finally we call the explainer for rows 0 and 2:

```
print("Threshold corresponding to Best balanced accuracy:
{:6.4f}".format(best_thresh))
show_explanation(0)
show_explanation(2)
```

**Cell Number: #20-26**
**Action to be taken:** Select cell numbers 20-26 and click RUN

You should get a similar result:



| Feature | Value |
|---|---|
| CHBRON=2 | True |
| COGLIM=2 | True |
| ADSMOK42=2 | True |
| PREGNT=-1 | True |
| MIDX=2 | True |
| EMPHDX=2 | True |
| CHDDX=2 | True |
| SOCLIM=1 | True |
| ANGIDX=2 | True |
| ADHDADDX=-1 | True |

```
Actual label: [0.]
```

See the LIME documentation for detailed description of results (https://github.com/marcotcr/lime).

In short, the left hand side shows the label predictions made by the model, the middle shows the features that are important to the instance in question and their contributions (weights) to the label prediction, while the right hand side shows the actual values of the features in the particular instance.

*Congrats!! You have finished the AI lab!*