**Introduction to Computer Science        CS105**

# Lab 8 : Machine Bias

## Deliverables

**Commit and push the following by 11:59 pm, Thursday, April 24, 2025**

Steps 3-6 can be done with your lab partner; you can work with one or two partners from class, with no restriction about working again with a partner you've worked with before. Feel free to arrange partners before lab, if you wish to do so.

1. The function **read_data** in the file **read_data.py**.
2. One or more analysis functions, with names you choose, in the file **analyze_data.py**. As per the description below, include a docstring, preconditions and tests (as examples in the docstring) for each function you write in this part.
3. The function **print_table**, and **code to call** it and your other functions, in **main.py**.
4. Your written answer to one of the questions that are listed here.

## Introduction

This week, we'll explore the topic of whether algorithms can "learn" from us, not only beneficial things (e.g., perhaps improving screening for tumors), but also harmful things, such as how to make racially biased decisions in the criminal justice system. In case it is not evident from this introduction, this topic is far more emotionally charged than most of the topics we discuss in CS105. If you feel that working on this lab might be problematic for you, contact your instructor to ask about whether we can find some other way for you to engage questions about how algorithms and computers impact society.

This lab will also give you a chance to build a meaningful program "from scratch". The starter files contain only the data to be analyzed, it will be up to you to create all the Python files and functions, based on the instructions below.

Our exploration will be a re-creation of the analysis done by ProPublica, on the data they gathered about an algorithm called COMPAS and about the outcomes of people who were affected by it. COMPAS is an algorithm designed by Northpointe that tries to predict the likelihood that a criminal defendant will reoffend in the future, or, in other words, their likelihood of *recidivism*. ProPublica is a nonprofit investigative journalism organization.

The base of this data set is the anonymous records of 7,000 people arrested in Broward County, Florida, in 2013 and 2014. Propublica gathered this data, and ran the original entries through COMPAS to obtain the predicted recidivism scores. They also recorded if the defendants were later arrested for committing another crime (within two years). All of this was compiled to make up the dataset provided in the starter files for your lab, and was used, together with additional investigation and analysis, in ProPublica's article about COMPAS; and an associated video from ProPublica (note that you may find some of the personal stories in the article upsetting, but the article is also where the history, analysis, and data are; you'll mostly get technical material if you skip to the 8th paragraph, which starts "Scores like this"). ProPublica found that the predictions of COMPAS were not only unreliable, but also consistently discriminated against black defendants.

This is not specifically covered in this lab, but note that Northpointe responded to ProPublica's article (here), and ProPublica responded to their response (here). Note that some of this discussion gets into additional statistical measures, such as sensitivity and specificity, which are defined in the P.S.U. document linked from Dave W.'s lecture slides.

You will repeat the analysis that separately compiles results for black vs. white defendants, to compare the relative outcomes for defendants with similar COMPAS ratings.

The programming steps for this lab are all to be done in the **MachineBias** project.

## 1. Recommendations for programming success

Unlike some previous labs, this lab is not designed to give you new *algorithm design* challenges; but, it is challenging in some other ways, including some pragmatic issues of programming. So, here are some techniques experienced programmers use, to try to take the easiest route to success. As with walking through woods or a city, the easiest path may not be a straight line, and programmers know how to take the easier but slightly longer "walk" around the programming equivalent of a bramble or crowded department store.

So, some suggestions (some of which are required elements of the lab; the suggestion here is just to do them *at the right time*):

- Write *and test* one function at a time, rather than writing everything and then starting to debug (if you've already started the lab by writing everything, you can still just try to debug one thing at a time *without use of your other functions*).

- Think carefully about the function parameters, and try to write them down before you write any code (use a comment if you've got a basic idea but don't know how to write it in Python). You may end up changing this type information, which seems like wasted work, but remember that we're looking for the *quickest and easiest* path, not the *shortest* path, to our answer.

- Test each function as soon as you write it, with a variety of small tests that include all the interesting cases you can think of. This, like the above, won't feel like the *shortest* path, but it helps avoid situations that are very slow and unpleasant.

- Whenever you read data, do your best to "validate" it, or at least the parts of it that you'll be using. For example, the recidivism column *should* have either a 0 or a 1; if your function reads a line that doesn't, it should complain somehow. Remember to test your input reader, especially the validator, and to use small tests, such as a couple of short .csv files you create yourself. Then, test on the big file that *should* have valid data, before going on to use the full data.

**NOTE:** don't try to print an entire huge data set on a terminal (including the PyCharm terminal) and then scroll back through it … terminals often have limited scroll-back, so you may not be able to see the full output by scrolling.

## 2. Before coming to lab session, prepare

Read the ProPublica article; your instructor may also have discussed this article in lecture

## 3. Create a function to read the data (with your partner)

The programming steps for this lab are all to be done in the **MachineBias** project.

The file compas_scores.csv contains a sequence of lines of data, each of which give information about one defendant. All are formatted in the same way, starting with sex/gender (defined and described in the limited terms used in the time and place the records were made), race (similarly), type of criminal charge (F for felony, M for misdemeanor (a lesser crime than Felony)), etc., with the first row giving the titles of the fields.

Create a file **read_data.py**, containing a function **read_data**, which takes one string giving the file name to be read, and returns the data in some appropriate format that captures the fact that the .csv file is a set of records with specific fields. You may use the python features you know, e.g., returning the result **as list of entries** (one for each row), with each entry being some data structure you feel is appropriate, e.g.

- each entry in your list could be a *dictionary* with keys like "sex" (that gives a value like "Male") and "two_year_recid" (that gives a value like "0");
- each entry in your list could be a *tuple* in which the 1st column is a string from column 1, etc., with each field having the appropriate type corresponding to what's in the CSV file
- each entry in your list could be a list of strings, each corresponding to one field in the file

Alternatively, if you know the *pandas* library, or would like to learn it through self-study (with some consultation with the faculty and T.A's), you may return the data as a *pandas data frame*, with the appropriate column headings. Note that faculty and T.A's may not be deeply familiar with pandas specifically.

Whichever you choose, describe the return type clearly and precisely, using either (a) Python's "typing" module, if you can, or (b) a comment, if you're unable to describe the return type with "typing".

We encourage you to do the following, but it is not required:

a. Make a small .csv file with some sample data, and make sure your function reads it and produces the appropriate output; you can include this example in a doctest comment.

b. Include some "data validation" steps, e.g., some fields only have a limited set of reasonable options, so if everything is going according to plan, you should always get, for example, either 0 or 1 for the recidivism column (titled two_year_recid). So, if you read any other value for that column, you should print a warning message that something is going wrong (you can choose to either raise an exception to stop the program, or just print a message about the problem, or both).

## 4. Create a function to analyze the data (with your partner)

Create a file **analyze_data.py**, and, in it, create one or more functions to get the various results shown in the figure "Prediction Fails Differently for Black Defendants". Each of these functions should take a parameter of the *same type* as what you return from your read_data function. You may write one function that returns all four percentages, or four functions that each compute one percentage.

Include appropriate **preconditions** and **docstrings** with **doctest examples** to demonstrate basic usage of each function you write, and to test that corner cases don't break your functions.

**Note:** *Your tests should use simple data sets you can enter by hand, for which you know the expected answer,* ***not*** *the full ProPublica data set!*

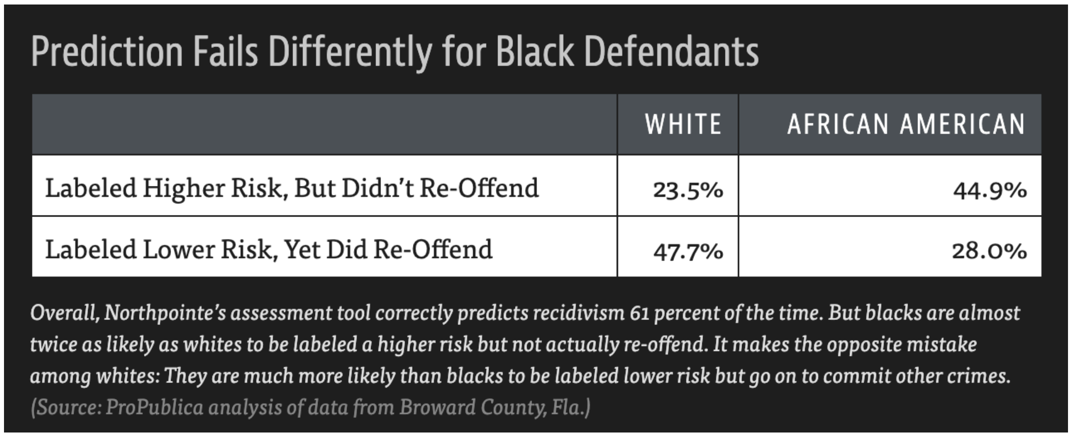
If you need help figuring out what is to be calculated, or how to calculate it, ask an instructor, or refer to this supplemental document or this video.

## 5. Create a function to print data, and a main program (with your partner)

Create a file **main.py**, and, in it, create a function **print_table** and write code **not in any function to call your functions in steps 2 and 3 and print the results**. In other words, when you click "Run" on main.py, it should just print the table (as text), like the one in the "Prediction Fails Differently for Black Defendants" article, by using your functions to read the data, analyze the data, and print the table from the four results (you can't just have a couple of print statements with the numbers from the article in them!).

For table formatting, we encourage you to use the tabulate package. The tabulate package has been installed on the machines in H110, so you should not have to install it (if you ever want to run this lab on your own computer, you'd need to install tabulate, by following the instructions at the link, and running "pip install tabulate" on the command line).

For reference, the table from the article is shown here, along with the definitions of the percentages:



## Prediction Fails Differently for Black Defendants

| | WHITE | AFRICAN AMERICAN |
|---|---|---|
| Labeled Higher Risk, But Didn't Re-Offend | 23.5% | 44.9% |
| Labeled Lower Risk, Yet Did Re-Offend | 47.7% | 28.0% |

*Overall, Northpointe's assessment tool correctly predicts recidivism 61 percent of the time. But blacks are almost twice as likely as whites to be labeled a higher risk but not actually re-offend. It makes the opposite mistake among whites: They are much more likely than blacks to be labeled lower risk but go on to commit other crimes. (Source: ProPublica analysis of data from Broward County, Fla.)*

*Figure 1: Relevant Percentages … Since there are different possible interpretations of the row labels above, see the box below for clarification. Note also that the table shows (100% - sensitivity), and (100% - selectivity), for each race, where sensitivity and selectivity are the "characteristics of the test." described in the PSU lesson; for those "characteristics", high numbers are good (like in baseball), so by subtracting the percentages from 100%, we can build the table above, in which high numbers are bad (like in golf).*

- Upper Row: Of the defendants of a specific race who **didn't re-offend,** what percentage was **labeled higher (i.e., other-than-low) risk**?
- Lower Row: Of the defendants of the specific race who **did re-offend,** what percentage was **labeled low risk**?

For those with a background in statistics, recall that we want a test to have good (high) *sensitivity* and good (high) *selectivity*; this chart gives the reverse of those measures, i.e., the rate of mis-predictions the test has, for each population. See this lesson for details.

## 6. Reflect on the context of this work (alone or together, your choice)

- Pick a question from the "Discussion Questions" appendix below, and write a paragraph or so with your thoughts. You are invited to write this by yourself, or to informally chat with your classmates as you form your opinions, or get a group of 2-3 students together to discuss it and develop some thoughts together (if you feel comfortable doing so).
- Submit your writing in a plain text (.txt) or PDF file, and add it to your project. Each student in a group should submit their own file, even if the group came up with the answer together.

## 7. Remember to submit your work

## Appendix: Discussion Questions.

1. The ProPublica article and resulting dataset that we will use for the lab indicate someone as having "recidivated" (committed another crime) if they were rearrested within 2 years. Specifically, the field indicating subsequent recidivism is stored as a binary 1 or 0.

   Exactly how is "recidivated" defined and measured? How might this data be inaccurate? How might it be biased or oversimplified? What might this mean about the resulting analysis?

2. COMPAS, the recidivism algorithm investigated by ProPublica, calculates a recidivism risk score based off of this questionnaire completed by the person being assessed. Consider the following questions that are asked:

   "Were your parents or siblings ever arrested, that you know of?"
   "How many prior drug possession/use offense arrests [have you received]?"
   "Is there much crime in your neighborhood?"[1]

   Though none of the questions directly mention race, how might they generate racial and socio-economic bias in the COMPAS algorithm? Consider factors such as over-policing of Black people in America and redlining of neighborhoods.

3. Many investigations like ProPublica's have been made into other decision-making algorithms for racial bias. For example, one algorithm designed to make decisions about medical care requirements underestimates the needs of Black patients. The engineers who designed the algorithm stated that they had not expected racial bias to emerge as it did, since they used supposedly "race-neutral" information like a patient's cost to the healthcare system.[2]

   How does our perception of certain kinds of data being race-neutral or color-blind influence public opinion about these kinds of algorithms? What impact might this have on regulatory policies for these algorithms? How do these algorithms perpetuate racial bias in data at a larger scale?

4. What was the original goal of the software developer(s) involved in this system? How did the developer in the article respond to requests to use this software for recidivism prediction? What else might a software engineer do when developing code like this, beyond supporting/ demanding separate auditing?

[1] *Sample Risk Assessment COMPAS CORE,* 2011
[2] Johnson, *Racial bias in a medical algorithm favors white patients over sicker black patients*