General guidelines:

- This assignment is tentatively due on **Wednesday of week 10,** with peer grades due during week 11. But final deadlines are **as posted on gradescope.** Note the Wednesday (rather than Sunday) deadline since everyone will need adequate time to perform peer grading during week 11.
- **Given that assignments must be distributed to peer graders, late submissions will not be accepted; there will be a late window though this is for students with an applicable OSD accommodation and not generally available to others.**
- Peer assignments will be available once the above late window closes
- Given that this is a new class, welcome to provide feedback on any problems with the spec!
- Assignments can be completed in groups of up to 4, though may also be completed individually
- The assignment will be peer-graded. You are welcome to "blind" your submission (to the extent possible) though you are not required to. *As such, there will be no difference in grading based on the group size.*
- You are required to upload:
    1. A **jupyter notebook**, exported as html. It should be "clean", documented, and readable as-is, i.e., the graders should not be expected to execute your code.
    2. A **~20-minute presentation**, uploaded as a video file (exact sharing instructions TBD).
    3. A peer grading report (due ~1 week later).
- **The above three components are worth 25% of your grade.** The notebook and presentation aren't graded separately, rather, *peer graders should be expected to watch your presentation while following along with your code.* The exact format is up to you, though I recommend a combination of slides and code walkthrough. The specific graded components are described below.

Overall description

In this assignment, you should analyze a dataset and its fairness characteristics using methods similar to those in class. You can use any dataset, but it must have:

- A label (*y*) (typically this will be binary, though you could also consider a regression task, or you could consider a single label in multiclass dataset)
- One or more sensitive attributes (*z*) (though these could be *any* binary attributes, and need not be gender/race/etc.). You should make sure there is some "unfairness" with regard to the chosen characteristics so that your results are interesting
- At least a few thousand samples and "plenty" of features (there are no specific guidelines here, though the dataset should be "large enough" that the methods from class are applicable)

Generally speaking you should *avoid using the exact datasets already covered in the class.* E.g. avoid using the COMPAS dataset where most of the assignment tasks are already covered by existing workbooks.

Your analysis should include the following components:

- ***Exploratory analysis and fairness metrics.*** You should report on:
    1. **Context:** Where does your dataset come from? What is it for, how was it collected, etc.?
    2. **Code:** Report standard accuracy and fairness metrics of labels and predictions, such as those in Modules 1 and 2;
    3. **Discussion:** Discuss which fairness metrics are specifically relevant to your task, e.g. accuracy parity might be less appropriate for recidivism prediction than demographic parity (etc.)
- ***Interventions.*** You should report on:
    4. **Code:** How much can "unfairness" in your predictions be explained by *dataset* characteristics? Can you fix them with dataset-based interventions?
    5. **Code:** How do different modeling choices impact fairness characteristics? Can you fix them with in-processing interventions?
    6. **Code:** Can you apply post-processing interventions to achieve desired fairness outcomes?
    7. **Discussion:** What types of interventions are most appropriate for your task (e.g. legal, practical to deploy, etc.)? What are the tradeoffs between them (e.g. how are other metrics negatively impacted by a particular intervention, etc.)
- Research paper. Implement a fairness intervention described in a research paper. Some possibilities include examples from class, e.g.:
    - *Data Pre-Processing Techniques for Classification without Discrimination,* Kamiran and Calders, 2011
    - *Fairness Constraints: Mechanisms for Fair Classification,* Zafar et al., 2017
    - *A Convex Framework for Fair Regression,* Berk et al., 2017

  or any appropriate paper *not* covered in class. Your should report on:
    8. **Context:** Summarize the main contributions of the paper and its relevance to your task.
    9. **Code:** Attempt to reproduce results similar to those reported in the paper on your dataset (or comment in detail about any failure to do so)
    10. **Discussion:** Is it more effective than other intervention strategies you tried? Why or why not? Conclude your presentation with a general discussion of what was and was not effective for your task.

It is ***strongly recommended*** that you follow the above section numbering and order, since graders will evaluate you on the above 10 elements. You are welcome to present the elements differently at your own discretion, just ensure that graders can easily find what's needed.

Grading

Graders will give a score out of 2 for each of the above 10 components. The following grading rubric should be used:

0: This component was not covered
0.5: This component was covered, but appears to contain errors, or cannot be easily understood, or the presentation does not seem to match the code, or the presentation seems to be automatically generated
1: This component was covered, but only superficially, i.e., it has some of the right components, and appears correct, but is missing key elements or comparisons that would be expected from a complete analysis
1.5: This submission is more-or-less minimally acceptable: e.g. a minimal set of essential metrics are covered, though maybe not discussed in detail
1.75: The code and presentation seems feature-complete: all the results "make sense", or any negative results are adequately explained; the presentation explains the tradeoffs between different metrics and goals, is clear and easy to follow, etc. ***This should probably be the most common grade.***
2: The code and presentation go above-and-beyond in some way, e.g. it might be particularly thorough, try an interesting or new approach, or shows excellence in some other way

You are welcome to use any real-valued grade in the range [0,2] for each of the 10 components.

Peer grading

Each grader will be expected to evaluate 5 assignments, worth 5 points of your grade. Your own grade for each of the above components will be the *median* of grades assigned by your peer graders.

Grade inflation

As this is a topics class running for the first time, I plan to have a fairly generous grade distribution. **But,** please grade assignments seriously, i.e., avoid giving full marks to every assignment. Although doing so might seem "nice", I receive a number of complaints from students who feel that their efforts relative to others are not recognized by peer graders. While I'm hesitant to implement an actual "curve" for this assignment, *if the grades in your report do not appear to be a genuine effort to grade the submitted assignment:*

- I may (silently) rescale your grades, e.g. if you grade every assignment 20/20 (or anything too near to that), I may multiply your grade by some constant < 1, such that you are actually penalizing your peers by entering "too generous" grades.
- Your 5 peer grading marks may be withheld.

Submission instructions

Unfortunately, (a) gradescope isn't set up for peer grading; (b) gradescope doesn't accept large files; and (c) google drive files can't be shared anonymously. So submission is a little complicated... It will work as follows:

Step 1: Submit your workbook and a *url path* to your video file:

- **workbook.html:** this should be a jupyter notebook, exported as html. *You should be able to open it in a browser.*
- **video_url.txt:** should contain a single line, which is a *path to a shared mp4 file on google drive.* It should probably look like: https://drive.google.com/file/d/FILE_ID/view?usp=drive_link

Probably, the latter should be generated on google drive by enabling "anyone with the link" sharing and copying the link. If you use some other format, please convert to mp4 (which can be generated by recording a zoom session or similar).

The autograder will check that (a) your file is downloadable; (b) greater than 1mb; (c) is of type video/mp4; and (d) that your workbook.html file is a html file.

Note that your google drive link *won't be shared:* it is not possible to fully anonymize shared files, so I will download your file and put it in a shared folder. Beyond that, it is up to you to what extent you'd like to anonymize your submission.

Step 2: About 1 day later, I will compile all submissions, make peer assignments, and distribute peer assignments via another gradescope assignment (details to follow).

How your submission is verified

In case you'd like to check at home, your submission is verified using the following code:

```
import magic
import requests

def test_workbook(path):
    f = open("workbook.html", 'r')
    t = f.read()
    if not t.startswith('<!DOCTYPE html>'):
        print("Your workbook submission is not a html file (does not start with
'<!DOCTYPE html>'")
        print("Please save your work as html (e.g. do not just change the extension)")
        raise Exception
    f.close()

def test_video(url):
    FILE_ID = None
    if "id=" in url:
        FILE_ID = url.split("id=")[1].split('/')[0]
    elif '/d/' in url:
        FILE_ID = url.split("/d/")[1].split('/')[0]
    if FILE_ID == None:
        print("Couldn't determine file ID; expected a url of the form
https://drive.google.com/file/d/FILE_ID/view?usp=drive_link or
https://drive.google.com/uc?export=download&id=FILE_ID")
        print('(got """' + url + '""")')
        raise Exception
    print("Looks like your google drive file ID is " + FILE_ID)
    url_fixed = "https://drive.google.com/uc?export=download&id=" + FILE_ID
    r = requests.get(url_fixed, stream=True)
    print("Trying to download from " + url_fixed)
    chunk = next(r.iter_content(chunk_size=1024*1024))
    if len(chunk) < 1024*1024:
        chunk = chunk.decode('utf8')
        if "Virus scan warning" in chunk:
            print("Looks like your file is too large for Google to scan for viruses")
            uuid = chunk.split('uuid" value="')[1].split('"')[0]
            url_fixed = "https://drive.usercontent.google.com/download?id=" + FILE_ID
+ "&export=download&confirm=t&uuid=" + uuid
            print("Trying to download from " + url_fixed)
            r = requests.get(url_fixed, stream=True)
            chunk = next(r.iter_content(chunk_size=1024*1024))
        else:
            print("Looks like your video file is less than 1mb; it is probably not a
video")
            raise Exception
    print("Confirmed that your video file is greater than 1mb; checking file type")
    mime = magic.Magic(mime=True)
    typename = mime.from_buffer(chunk)
    if not "video/mp4" in typename:
        print("Your file is not an mp4 video file (type=" + typename + ")")
        raise Exception
    print("Everything looks okay! On your own, please verify that you can download a
working video using:")
    print("wget -O output.mp4 '" + url_fixed + "'")

test_workbook("workbook.html")
f = open("video_url.txt", 'r')
url = f.read().strip()
test_video(url)
f.close()
```