Assignment 6 (web) (INF3331: 30 + 15 points, INF4331: 40+5 points)

All solutions should be stored in a directory called `assignment6` in your private repository.

6.0: Background (0 points)

In this assignment, you will be building a web-based visualization of temperature and CO_2 data. The data sets are available in `student-resources-16` as `CO2.csv` and `temperature.csv`, and are taken from `http://berkeleyearth.lbl.gov/regions/contiguous-united-states` and `http://cdiac.ornl.gov/trends/emis/meth_reg.html`.

6.1: Temperature/CO_2 plotter (10 points)

Build a Python script which reads data from the files `CO2.csv` and `temperature.csv` and generates a labeled, nice plot of time vs. CO_2 or time vs. temperature. It should implement methods `plot_temperature()`, `plot_CO2()`. The user of those methods should be able to control at least the following:

- For temperature: which month to plot temperatures from, for
- Time range to plot
- y-axis min, max

so make sure your methods take appropriate arguments. Make sure to add docstrings to your methods describing which arguments do what.

Name of file: `temperature_CO2_plotter.py`

6.2: Visualization web app (6 points)

Build a Flask app which uses your script from 6.1 to generate a plot of temperature and a plot of CO_2 and display it on the web page.

Name of file: `web_visualization.py`

6.3: Interactive visualization (8 points)

Modify your solution in 6.2 so that the visitor of the web page can change the parameters of the plot using drop down menus, radio buttons or whatever reasonable option you prefer.

6.4: Visualization of CO_2 emissions by country (10 points)

Using the data from the file `CO2_by_country.csv`, extend your script from 6.1 with a method which takes as input an upper/lower threshold, and generates a bar chart of the CO_2 emissions of all countries with per capita emissions above/below that threshold. Next, extend your script from 6.2 so that a plot of this is also generated, and the user is given the option to specify upper/lower thresholds on the web page as well.

6.5: Documentation and help page (6 points)

Extend your web app with a help-page, and add a link to this on your plot page. The help page should display help for your methods in `temperature_CO2_plotter.py` generated automatically from docstrings using your favorite tool for this. Options here include pydoc or Sphinx.

6.6: Predicting the future (5 points)

Add an option to your temperature visualization to predict the temperature in the future, and plot it. Make sure to use a line visually different from the actual observed temperature to make it clear to the user that you are interpolating.

Doing proper climate modeling is hard, so you may simlpify this heavily as follows: assume that temperature is roughly a linear function of CO_2 emission, estimating the coefficients of the linear function from recent data points (using the past 2 is fine, as is using the past 10 or so if you want to be more thorough).

Further, assume that the rate of increase of CO_2 emissions is going to be the same as it is today (i.e. if there were X tons more CO_2 emissions in 2016 than in 2015, there will be X tons more CO_2 emissions in 2017 than in 2016). Using this, you will be able to get an estimate of the CO_2 emissions and temperature in later years.

Finally, add an option to your web page to let the user specify how far into the future you want to plot.

Clarifications

- Doing proper climate modeling is hard, and due to the nature of the subject, hard to make completely apolitical. It is not our intention to force any political beliefs upon you.
- Note that the CO_2 2 data set in 6.4 is taken from a different source than the one in 6.1. As such, there may be discrepancies between them.
- If a later exercise asks you to add functionality to a previous exercise, you do not have to submit both the old version and the updated version - it is fine to just submit the updated version.