CS 251 OUTLAB 4 - Numpy, Scipy, Matplotlib, Pandas

General Instructions

- Make sure you know what you write, you might be asked to explain your code at a later point in time.
- The submissions will be graded automatically, so stick to the naming conventions strictly.

Submission Instructions - Your folder structure should be like this.

```
<rollno1>-<rollno2>-outlab4/
│
├─── task1/
│          ├─── P1/
│          ├───      │
├───             ├───      └─── analysis.py
├───             ├─── P2/
├───             ├───      │
├───             ├───      ├─── convert.py
├───             ├───      └─── transformed.csv
├───             └─── P3/
├───                       │
├───                       ├─── merge.py
├───                       └─── info_combine.csv
├─── task2/
├───             │
├───             ├─── task2.py
├───             ├─── instance1.png
├───             ├─── instance2.png
├───             └─── instance3.png
├─── task3/
├───             │
├───             ├─── task3.py
├───             ├─── Lenna_2.png
```

```
├──────            ├──── Lenna_3.png
├──────            ├──── Lenna_5.png
├──────            └──── Lenna_10.png
├──── task4/
├──────            │
├──────            ├──── task4.py
├──────            ├──── fn1plot.png
├──────            ├──── fn2plot.png
├──────            ├──── bd_1.png
├──────            └──── bd_2.png
├──── task5/
├──────            │
├──────            ├──── task5.py
├──────            └──── covid.png
└──── references.txt
```

Your submission directory should be called <rollno1>-<rollno2>-outlab4. These should be sorted lexicographically.
For example:
17D070059-180070035-outlab4

Compress it into a tarball, strictly with the following command:
`tar -zcvf 17D070059-180070035-outlab4.tar.gz 17D070059-180070035-outlab4`
Of course, replace our roll numbers with yours!

You have to submit on Moodle. The link is up. Submit once per pair, from the account of the student with the lexicographically smaller roll number, i.e. rollno1. In our example, 17D070059 will submit.

Deadline: 13th September, 2020 11:59pm.
After that 2^H % penalty for each H hours late, upto 6 hours. After that, no submission is allowed. Mind it, even 1 minute extra counts as an hour late.

Task 1 : Numpy (15 marks)

You are given a CSV file mumbai_data.csv, which contains information about Testing, Infected, etc. The file has the following fields -
Day, Tests, Infected, Recovered, Deceased
The data has been collected in the morning.
Your job is to complete the following tasks -
****
P1
****
Create **analysis.py** to print the mean and standard deviation values for each field on **stdout**.
Store the data from the CSV file into a NumPy array and calculate the mean and standard deviation.
The format should be as follows -

Field	Mean	Std. Dev.
Tests	…	…
Infected	…	…
Recovered	…	…
Deceased	…	…

****
P2
****
Create **convert.py** to convert the Tests from Total tests to Tests per Million and Infected to Test Positivity rate. The total population of Mumbai is 20.4 million.
Read **mumbai_data.csv**, convert it to a NumPy array and perform the above operation.
Create another CSV file **transformed.csv**
The format will be the same as mumbai_data.csv, convert Tests as Tests per Million and include Test Positivity rate in place of Infected
****
P3
****
New data has been collected which includes the information observed after the lockdown was lifted. This data is stored in mumbai_unlock.csv
Create **merge.py** to combine mumbai_data.csv and mumbai_unlock.csv.
Name the merged file as **info_combine.csv**
The format should be as follows -

Day	Infected(Unlock)	Infected(Lock)	Positivity Rate(Lock)	Positivity Rate(UnLock)
Monday	…	…	…	…

Task 2 - Pandas (25 marks)

The file sml.csv contains 1 header + 7350 rows of data (total 7351 lines). It has 7 columns (as mentioned in the header): instance, algorithm, randomSeed, epsilon, horizon, REG.

You will generate three plots: one for each instance.

The plot will have “horizon” on the x-axis and “regret” on y-axis. Use log-scale on both axes. It will contain 7 lines: one for each algorithm (counting epsilon-greedy as three algorithms). For other algos, ignore epsilon. Each point will give the **average regret** from the **fifty random runs** at the **particular horizon** for the algorithm. Make sure you provide a clear key (legend) so the plot is easy to follow. Your plots should contain titles on both axes as well as on the main plot.

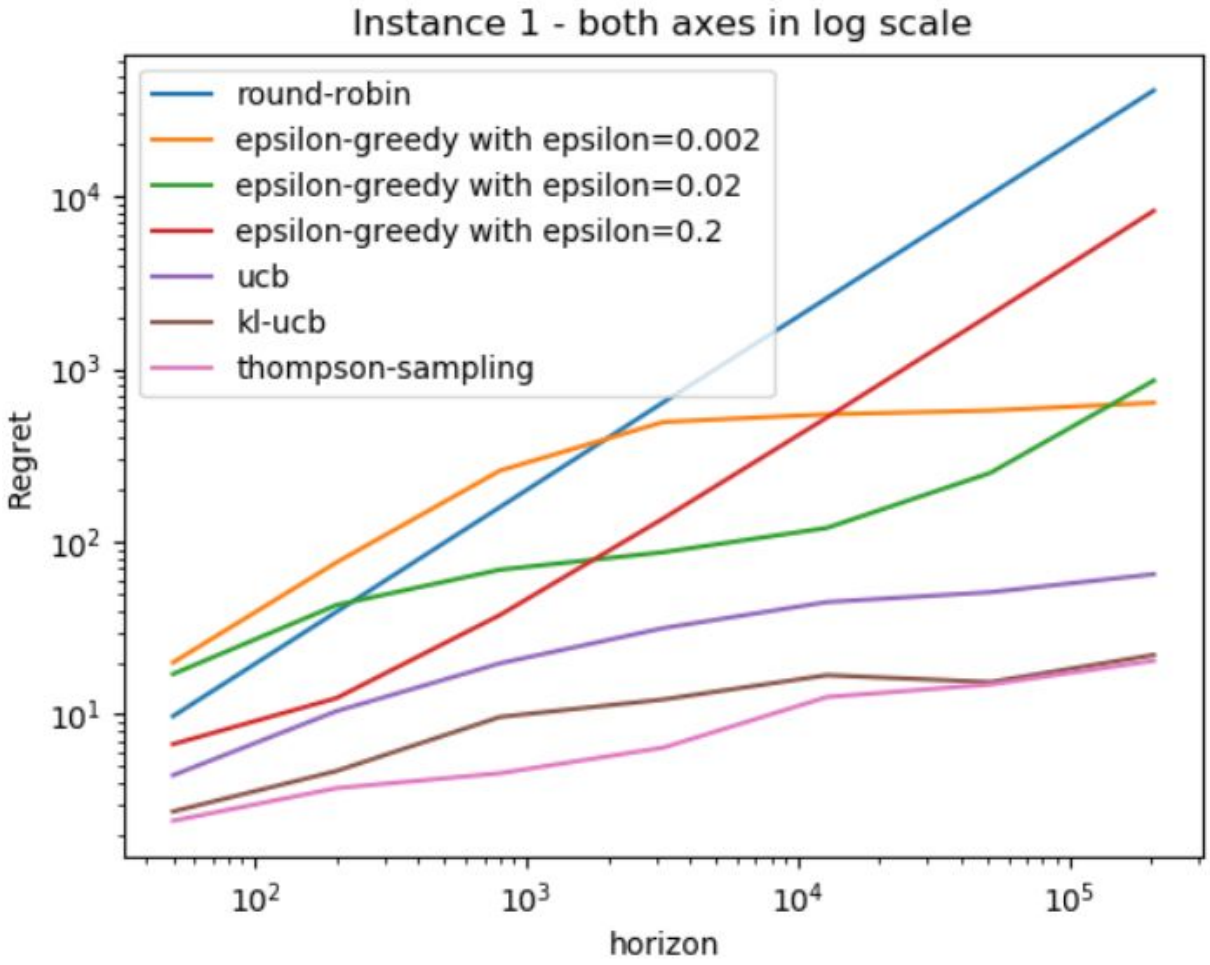
Name your script as task2.py. We should be able to call it as:

```
python3 task2.py --data sml.csv
--data: is the path of data file
```

Your script should produce three plots:
instance1.png, **instance2.png** and **instance3.png**. Note that these three plots should be present in your submission directory. (We will be testing your script with other csv files too, so be careful!)
Important : Plot using **Matplotlib.**

ProTip: Pandas will be really helpful here. See **Resources**.

Following is a sample plot
Edit 1 : Note : Don't focus on your plotting exactly matching the output we provided. It's from a different file. It's there only so you know the **formatting** of your output. Even the colors can be different, no worries
Edit 2 : Legend ordering does not matter either.

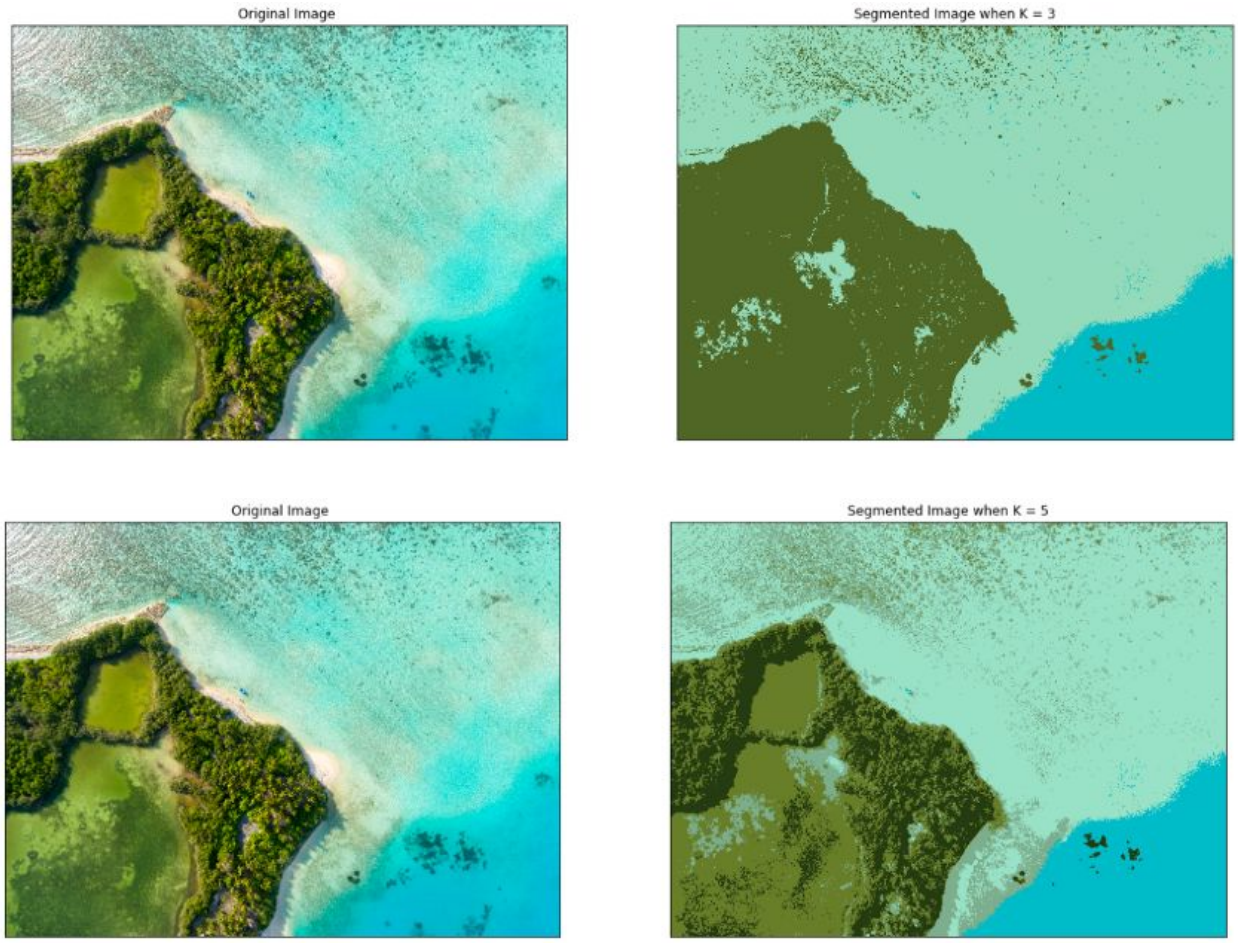


Helpful Resources :

1. https://www.datacamp.com/community/tutorials/argument-parsing-in-python
2. https://queirozf.com/entries/pandas-dataframe-plot-examples-with-matplotlib-pyplot
3. https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.groupby.html
4. https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.sample.html

Task 3 : Scipy (20 marks)

Coloured images are represented as 3D arrays of Red, Green and Blue values, where each vector is a pixel. Multiple algorithms exist, which can group these vectors together based on a distance metric. We will focus on KMeans++. You give it the vectors and a number k , which denotes the number of such groups. Without delving into how the algorithm works, just look up [scipy.cluster.vq.kmeans2.]



Source : [Introduction to Image Segmentation with K-Means clustering]

Problem Statement

Write a script called task3.py, which takes following commands:

--input in, where in is the path of the input image.
--k k, k in KMeans++. It's an integer, ranging from 1 to 50.
--output out, where out is the path of output image

It will take the image, apply KMeans++ algorithm on its pixels and replace each pixel by the centroid of the group of its pixel and save the constructed image back.

You can use whatever you want to complete this task (loops are also allowed). We will just see the output image. Don't bother about checking for invalid inputs.

Here is a sequence of **recommended steps to do this task**. Using some other way is also fine.

1. Read the image into a NumPy array. The array elements will be integers in range [0,255]. You should convert it into float.
2. Convert the array into a 2d-array of shape (n, 3). n (= n_r * n_c) is the total number of pixels in the image. n_r and n_c are the number of rows and columns in the image. Let's call this matrix M. Each row of M is a vector representing a pixel in an image. Again, the vector contains the RGB values at that pixel.
3. Pass this matrix to kmeans2 with the argument **minit = '++'**.
4. KMeans++ will give you two return values:
    1. Centroid: an array of shape (k,3). i^{th} index is the centroid of i^{th} cluster.
    2. Label: label[i] is the index of the centroid which is closest to i^{th} row of M.
5. Use this to replace each pixel by its centroid.
6. Save the image back to the output path.

Use the provided image Lenna.pg and submit Lenna_2.png, Lenna_3.png, Lenna_5.png and Lenna_10.png along with task3.py. Lenna_<i>.png indicates k=i.

Task 4 : Numpy, Scipy, Matplotlib (25 marks)

- Write a function `fn_plot1d` which takes a single argument function `fn`, along with a finite continuous domain and a filename, plots that function and saves it into a file. You can assume that `fn` is well behaved in the given domain.
  The signature of `fn_plot1d` should be:
  `def fn_plot1d(fn, x_min, x_max, filename):`
- Use `fn_plot1d` to plot `b(x)`:
  $$b(x) = g(|x|)$$
  where:
  $$g(x) = \frac{h(2-x)}{h(2-x) + h(x-1)}$$
  where,
  $$h(x) = e^{\frac{-1}{x^2}}, \ if \ x > 0$$
  $$0 \ otherwise$$
  in domain [-2,2]. Save it into file `fn1plot.png`

- Similarly write `fn_plot2d` which takes a two-argument function `fn`, along with a rectangle and a filename, creates a surface plot of that function and saves it into a file. Again, assume that `fn` is well behaved in the given domain.
  The signature of `fn_plot2d` should be:
  `def fn_plot2d(fn, x_min, x_max, y_min, y_max, filename):`
- Use `fn_plot2d` to plot the 2d sinc function:
  $sinc(x,y) = \frac{sin(\sqrt{x^2+y^2})}{\sqrt{x^2+y^2}}$ if $\sqrt{x^2+y^2} > 0$
  1 *otherwise*
  In domain $x, y \in [-1.5\pi, 1.5\pi]$. Save it into file `fn2plot.png`

  Note: You **CAN'T** use loops to implement both `fn_plot1d` and `fn_plot2d`.
  ProTip: Lookup `linspace, meshgrid, mplot3d, vectorize`

- Create a function `nth_derivative_plotter` which takes following arguments:
  - `fn`: a well behaved function in the given domain
  - `n`: a positive integer. fn should be n-times differentiable.
  - `xmin`: lower limit of domain
  - `xmax`: upper limit of domain
  - `filename`: name of the output file

  `nth_derivative_plotter` should plot the n^{th} derivative of `fn` in the given range and save it into a file with name `filename`. It should also add a suitable title to the plots! You have to **implement it without any kind of loops**.
- Use `nth_derivative_plotter` to plot the first and second order derivatives of $b(x)$ in [-2,2]. Save nth derivative with name `bd_<n>.png`. For example, plot of 1st derivative should be saved as `bd_1.png`. You **can** use loop for this part.
  ProTip: lookup `scipy.misc.derivative`

  All plots **MUST** have labels for all axes and title for the graphs

Submit `task4.py`, along with `fn1plot.png, fn2plot.png, bd_1.png and bd_2.png`

Task 5 : Levitt's Metric (25 marks)

Metric : $H(t) = \frac{x(t)}{x(t-1)}$,

where x(t) is cumulative count until day t, of covid deaths.
So, H(today)= (Total deaths till today) / (Total deaths till yesterday).

It is proposed that when H(t) = 1, then the pandemic is over.

So, you are required to fetch the Covid data from the link provided, plot the H(t) vs t graph and approximate the end of the pandemic in India using linear regression over H(t) and finding when H(t) =1 using the line obtained. Use data starting from **15 April.** Your program should output a single integer, the number of days it will take for the pandemic to end, taking 15 April as 1, as well as plot the line obtained using linear regression on the same graph as above. Provide proper axis labels, title and legend. The program should save the graph as covid.png. You can use pandas for processing the CSV file.

CSV file : COVID 19 India data
Pro tip: Lookup fetching CSV files using python requests module, scipy.stats.linregress. Use pip to install requests module, if not available in Docker container.
~
Submit task5.py and covid.png

Fun fact: India is a huge country with diverse demographics. You can collect the data of your district to predict when will the pandemic end in your district. :p