IE 582 Statistical Learning for Data Mining
Homework 1, due October 29th, 2021

Instructions: Please solve the following exercises using R (http://www.r-project.org/) or Python (https://www.python.org/). You are expected to use GitHub Classroom and present your work as an html file (i.e. web page) on your progress journals. There are alternative ways to generate an html page for you work:

- A Jupyter Notebook including your codes and comments. This works for R and Python, to enable using R scripts in notebooks, please check:
  - https://docs.anaconda.com/anaconda/navigator/tutorials/r-lang/
  - https://medium.com/@kyleake/how-to-install-r-in-jupyter-with-irkernel-in-3-steps-917519326e41

  Things are little easier if you install Anaconda (https://www.anaconda.com/). Please export your work to an html file. Please provide your *. ipynb file in your repository and a link to this file in your html report will help us a lot.

- A Markdown html document. This can be created using RMarkdown for R and Python Markdown for Python

Note that html pages are just to describe how you approach to the exercises in the homework. They should include your codes. You are also required to provide your R/Python codes separately in the repository so that anybody can run it with minimal change in the code. This can be presented as the script file itself or your notebook file (the one with *.ipynb file extension).

The last and the most important thing to mention is that academic integrity is expected! Do not share your code (except the one in your progress journals). You are always free to discuss about tasks but your work must be implemented by yourself. As a fundamental principle for any educational institution, academic integrity is highly valued and seriously regarded at Boğaziçi University.

Task 1 – Curse of dimensionality and effect of sample size

This problem looks at a simple simulation to illustrate the curse of dimensionality. First you'll compare the volume of a hypersphere to a hypercube as a function of dimension. Then you'll look at the average distance to nearest points as a function of dimension.

For D from 1 to 15 dimensions, simulate 1000 random D-dimensional points, where the value in each dimension is uniformly randomly distributed between -1 and +1.

a) Calculate the fraction of these points that are within distance 1 of the origin, giving an approximation of the volume of the unit hypersphere to the hypercube inscribing it. Plot this fraction as a function of D (a scatter plot of D versus the fraction). Note that Euclidean distance between two points, p and q, in D-dimensional space is defined as

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + ... + (p_i - q_i)^2 + ... + (p_n - q_n)^2}.$$

where p_i is the i^{th} dimension of point p and q_i is the i^{th} dimension of point q.
(check http://en.wikipedia.org/wiki/Euclidean_distance if not clear)

b) Use the value of this fraction at $D = 2$ and $D = 3$ to get estimates for the value of pi (Π) as you know the area (for $D = 2$) and volume (for $D = 3$) formulae for these cases.

c) Perform the calculations in part (b) with larger sample sizes. You can use the following set: {5000, 10000, 25000, 50000, 100000}. Visualize the estimated Π for $D = 2$ and $D = 3$ cases. Comment on your results

d) Repeat this simulation, sampling 1000 D-dimensional points from 1 to 15 dimensions, where the value in each dimension is uniformly randomly distributed between -1 and +1. For each value of D, generate an additional 100 test instances and calculate the distance to each test instance's nearest neighbor. Plot the average distance from the test instances to their nearest neighbors as a function of D.

Some references (in case you need)
hypersphere: http://mathworld.wolfram.com/Hypersphere.html, http://en.wikipedia.org/wiki/N-sphere
hypercube: http://en.wikipedia.org/wiki/Hypercube

Task 2 – Practicing data manipulation skills on images

In this exercise, you are requested to perform certain operations on images. The aim is to develop data manipulation skills necessary for this course.

Here is a background information about how a grayscale image is represented on our computers. A grayscale image is basically a matrix where each matrix entry shows the intensity (brightness) level. In other words, when you take a picture with a digital camera, the image is represented by a numerical matrix where the matrix size is defined by the resolution setting of your camera. If your resolution setting is 1280x720, then your image is represented by 1280x720= 921600 pixel values (Actually that is why higher resolution provides better quality pictures). When you have a color image, the image stores the information of multiple channels depending on the image type. The most famous one is RGB type where R, G and B stand for "red", "green" and "blue" respectively. Hence, you have a matrix as in greyscale images representing the intensity for each channel. Combining these matrices generates the color image.

Below is the steps you need to follow for this task:
- Take a picture of yours and save it as *.jpg or *.jpeg file.
- Using an image editor (i.e. Paint in Windows), crop your head from the image and save it as separate image file (again *.jpg or *.jpeg). The resulting image is expected to be a square (i.e. has the same length and width), hence you should crop the image accordingly.
- Resize the cropped image to size 512x512 px (pixel) using an image editor. This image is the one that you will use for this exercise.

a) Read image as a variable in R/Python. You need to install “jpeg” package to read image into a variable if you use R. For Python, an alternative is to use matplotlib package. What is the structure of the variable that stores the image? What is the dimension? a. Display the image. (Hint: google “rasterImage”)

b) Display each channel as separate image

c) For each channel, take the average of the columns and plot the average as a line plot for each channel on a single plot.

d) For each channel, subtract one half of the image from the other half (choice of halves is up to you but dividing the head image vertically into two parts make more sense). If you observe negative pixel values, you can make them equal to zero. Then:

- Display the new image.
- Display each channel separately as separate image.

e) In order to create a noisy image, add a random noise from uniform distribution with minimum value of 0 and a maximum value of “0.1 * maximum pixel value observed” to each pixel value for each channel of original image.

- Display the new image.
- Display each channel separately as separate image.