IC152: Assignment 8
Pandas, Images and Binary Search

3 questions **60 marks**

You must keep filenames as mentioned in the assignment since the assignments will be auto evaluated via a script. There are 3 problems , the first is for 20 marks, and the next two for 15. 10 marks are for viva. VERY IMPORTANT: Save all python files and outputs in a single folder. You must name python files as suggested in each question. Do not miss writing your roll number in the folder name of the zip file. Example folder name: b22000_assignment8.zip.

Problem 1: Pandas

Save the code in this question for all parts in file named problem1.py

The Cars93 dataset contains data from 93 cars that were on sale in the USA in 1993. The dataset has 93 rows listing various car models along with their features and specifications.

The columns of this dataset---there are 27 of them---comprise various attributes, e.g., Manufacturer, Type, Price, mileage, horsepower etc. Refer to the Appendix (at the end of this assignment) for more information.
....

# Getting started with pandas:
https://pandas.pydata.org/docs/getting_started/index.html
# API reference: https://pandas.pydata.org/docs/reference/index.html
# Reading the data from a csv file and storing it in a pandas dataframe
import pandas as pd
df = pd.read_csv("data/Cars93.csv")
....
....
Open data/Cars93.csv and observe its contents.

a. List all the models mentioned in the Cars93 data set in alphabetical order. Save them in ‘problem1a.csv’ with 2 columns: column 1 should have row numbers like input file, and column 2 should have sorted Manufacturer names.
   Hints (print each, then choose the appropriate ones):
   df.sort_values(['col1', 'col2'])
   df.sort_values('col1', ascending=False)
   df['col1'].sort_values()
   df['col1'].unique()
   df.to_csv('problem1a.csv')

b. Print all details of the costliest car of each of the ‘Types’.
   Hints (print each, then choose the appropriate ones):
   df.groupby('col_i')
   df.groupby('Type').get_group('small')
   df[df['col_i'] == df[‘col_i’].max()]

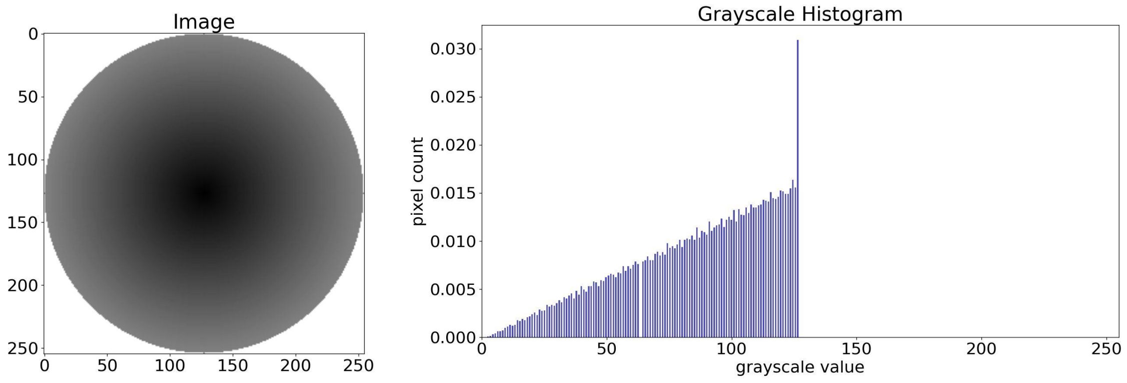
c. Write a function that asks you to enter the name of a manufacturer and print all the models produced by that manufacturer. Take the

first command line argument of ‘problem1.py’ as the name of the manufacturer.
Hint: df[df[‘col1’]== ‘string1’]

d. Write the total count of cars per manufacturer in ‘problem1d.csv’ in the same format as that of problem1a.csv but with an extra column with heading ‘count’.
Hints:
df.groupby('col1').size()
df.reset_index(name='col2')

Problem 2: Image Creation and Image Histograms

a. Run problem2.py and try to understand it from comments.
b. Modify it to create the following figure and histogram (not bar at value 255 should be carefully removed from histogram). Save a python file with name problem2b.py and a common image having both subfigures with name problem2b.png.



c. Now read the `'problem3cInput.jpg'` image using the "helper code" given below and create the histogram of:
   i. Its pixel values
   ii. difference between each pixel with a pixel left to it (loop for column number from 1 to end). What do you observe? Share with the TA/instructor.

# helper code starts:

```
    # install pillow using:
    # pip3 install pillow
    # if above does not work and you are on windows,
try:-
    # py -m pip3 install pillow

    # importing Image and ImageOps from PIL
    from PIL import Image, ImageOps

    # creating image object:
    imInp = Image.open('problem3cInput.jpg')

    # converting imInp to grayscale:
    imGrayscale = ImageOps.grayscale(imInp)

    # converting image to numpy array:
    import numpy as np
    pixels2D = np.array(imGrayscale)
    print(pixels2D.shape)

    # showing/saving image
    imGrayscale.show()
    #imGrayscale.save('problem3cOutput.jpg')
```

# helper code ends.

Save your python file as problem2c.py, and running it should save two images with names problem2ci.png and problem2cii.png.

Problem 3: Binary Search

If you have a sorted array/list, you can apply binary search to quickly find a value's index in the array/list by i) comparing the middle value of the array with the value being searched and ii) sub arrays on its right or left depending on the middle value.

- E.g. if you want to search 15 in [3, 5, 7, 9, 11, 13, 15],
- Then you will compare 15 with the middle value i.e. 9.
- Now since 9 is less than 15, the next search will be in values right to 9, i.e. among the last three elements. (If we would have got the value in the middle as greater than 15 we would have continued searching in the left part, for the middle value equal to 15 we would have stopped).
- Now the middle value of the right part to 9 ([11, 13, 15]) is 13, which is less than 15. So we will search in the right part again.
- Now the middle value of part right to 13 ([15]) is 15 as only one element is left, and it is the equal to the value being searched. So, now we know the location of 15 and we stop our search.
- Observe that in 3 comparisons we have reached the last value. If we would have used a loop from 1 to n (7 in this case), we would have to do 7 comparisons. Hence binary search is the fastest way to search the index of a value in a sorted array.

a. Write a function to apply binary search on any generic input list and save it as problem3.py. The function should return the index of the value in the input array. The code should be named as problem3a.py

and it should take filename with command line argument. The input file should have following content (for example) with first line having search value and second line having the array:
20
1 8 20 100 139 212 302 53113 1211

b. Write a code using a for loop.

c. Estimate the time problem3a.py and problem3b.py take for different sorted lists with around 100000 elements and share your observation with TAs/instructor.

   Note: Time can be estimated by importing time library as follows:
   import time start = time.time()
   #execution of the function
   end = time.time()
   total_time = (start-end)

Create the folder having your python files and inputs/outputs, with name having your roll number followed by "_assignment8" (don't use inverted commas in folder name), compress the folder <u>with .zip extension</u> and submit it on moodle.

Important: If you copy the assignment or any of its parts from others or share with others, our plagiarism softwares will catch it and you will be awarded 0 marks for the whole assignment or F grade for the course.

Appendix

Source:
https://stat.ethz.ch/R-manual/R-devel/library/MASS/html/Cars93.html
(retrieved on 31 Dec'22)

Details:

Cars93 {MASS}	R Documentation

Data from 93 Cars on Sale in the USA in 1993

Description

The Cars93 data frame has 93 rows and 27 columns.

Usage

Cars93

Format

This data frame contains the following columns:

Manufacturer

Manufacturer.

Model

Model.

Type

Type: a factor with levels "Small", "Sporty", "Compact", "Midsize", "Large" and "Van".

Min.Price

Minimum Price (in $1,000): price for a basic version.

Price

Midrange Price (in $1,000): average of Min.Price and Max.Price.

Max.Price

Maximum Price (in $1,000): price for “a premium version”.

MPG.city

City MPG (miles per US gallon by EPA rating).

MPG.highway

Highway MPG.

AirBags

Air Bags standard. Factor: none, driver only, or driver & passenger.

DriveTrain

Drive train type: rear wheel, front wheel or 4WD; (factor).

Cylinders

Number of cylinders (missing for Mazda RX-7, which has a rotary engine).

EngineSize

Engine size (litres).

Horsepower

Horsepower (maximum).

RPM

RPM (revs per minute at maximum horsepower).

Rev.per.mile

Engine revolutions per mile (in highest gear).

Man.trans.avail

Is a manual transmission version available? (yes or no, Factor).

Fuel.tank.capacity

Fuel tank capacity (US gallons).

Passengers

Passenger capacity (persons)

Length

Length (inches).

Wheelbase

Wheelbase (inches).

Width

Width (inches).

Turn.circle

U-turn space (feet).

Rear.seat.room

Rear seat room (inches) (missing for 2-seater vehicles).

Luggage.room

Luggage capacity (cubic feet) (missing for vans).

Weight

Weight (pounds).

Origin

Of non-USA or USA company origins? (factor).

Make

Combination of Manufacturer and Model (character).

Details

Cars were selected at random from among 1993 passenger car models that were listed in both the *Consumer Reports* issue and the *PACE Buying Guide*. Pickup trucks and Sport/Utility vehicles were eliminated due to incomplete information in the *Consumer Reports* source. Duplicate models (e.g., Dodge Shadow and Plymouth Sundance) were listed at most once.

Further description can be found in Lock (1993).

Source

Lock, R. H. (1993) 1993 New Car Data. *Journal of Statistics Education* **1**(1). doi:10.1080/10691898.1993.11910459

References

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S-PLUS.* Fourth Edition. Springer.