COMP20008 Project 1

V1.0: 20th March 2019

Due Date

The assignment is worth 20 marks, worth (20% of subject grade) **and is due 11:59pm 10th April 2019**. Submission is via the LMS. Please ensure you get a submission receipt via email. If you don't receive a receipt via email, this means your submission has not been received and hence cannot be marked. Late penalty structure is described at the end of this document.

Objectives

The objectives of this assignment are

- To practice a selection of cleaning and visualisation techniques discussed in lectures and workshops.
- To practice using a widely used Python library for data processing - *pandas*.
- To gain experience using library functions which are unfamiliar and which require consultation of additional documentation from resources on the Web.

Background

In this project you will be practice your Python wrangling skills with a publicly available dataset. The dataset is the food nutrient database from Food Standards Australia and New Zealand[1] in the file *food_nutrient_2011_13_AHS.csv*. It contains a detailed breakdown of 51 nutrient values for a wide range of food items. Each nutrient value is presented on a per 100 gram edible portion basis. It was used to support the 2011-13 Australian Health Survey. Table 1 shows a high level summary description of each attribute.

There are two data files provided on the project page on the LMS:

- *food_nutrient_2011_13_AHS.csv:* Nutrient information for about 5.7k foods. Used in all questions.
- For Question 9 only, you will also use an additional dataset, *8i. Australian Health Survey Classification System.csv*

[1] http://www.foodstandards.gov.au/science/monitoringnutrients/ausnut/ausnutdatafiles/Pages/foodnutrient.aspx

Field Name	Description
Food ID	8 digit alpha numeric food identification code, based on FSANZ standard
Survey ID	8 digit food identification code that was specific to Australian Health Survey
Survey flag	can be ignored
Food Name	Name of the food
nutrient name (units)	Describes the full nutrient name of each of the nutrients e.g. 'Protein' and includes the units the nutrient is presented in e.g. '(g)' for grams. A value is then provided for each food and nutrient.

Table 1: Summary of Attributes for *food_nutrient_2011_13_AHS.csv*

Libraries to use are Pandas and Matplotlib. You will need to write Python 3 code and work with Series and DataFrames discussed in workshop week 2 and data cleaning and basic visualisations in workshops weeks 3-5. If you are using other packages, you must provide an explanation in your code about why it is necessary.

Importing required Python Libraries and loading the data

Please begin your Jupyter notebook by listing all the Python libraries you will be using. Also load the dataset (.csv) in a dataframe object.

```
#import ....
import pandas as pd
import numpy as np
food =  pd.read_csv("food_nutrient_2011_13_AHS.csv", header=0,low_memory=False)
```

1 Dataset Summary (1 mark)

Print the number of foods, number of attributes, attribute names and their datatypes. The output of this step should look like

```
***
Q1
Number of foods: #
Number of attributes: #

@ $
@ $
@ $
...
@ $
***
```

where # is the value you find, @ is an attribute name and $ is its datatype.

Hint: Useful functions are shape and dtypes

2 Basic statistics (1 mark)

What is the median value of *Magnesium (Mg) (mg)*? What is the mean value of *Moisture (g)*? Your code should print out the results with the following format:

```
***
Q2
Median value of Magnesium: #
Mean value of Moisture: #
***
```

where # is the value you find, rounded to 1 decimal place.

Hint: Useful functions are mean and round

3 Data transformation adding a new attribute to the table (2.5 marks)

a) Convert the datatype of the *Survey ID* attribute into the string type.

b) Using *Survey ID*, create a new attribute *Food category*, which contains the first two digits of the attribute *Survey ID*.

c) Using this new attribute, compute the number of foods with *Food category* equal to '13' (this category can be described as "Cereal based products and dishes"). Using this information, now print out the percentage of foods which are in this category.

The output of this step should look like

```
***
Q3
% of foods which are Cereal based products and dishes (Food category 13)  = #
***
```

where # is the value you calculate rounded to 1 decimal place.

Hint: Useful functions are astype and string slicing and round

4 Visualisation using boxplots (2 marks)

Draw a plot consisting of two boxplots. One boxplot to show the distribution of *Total fat (g)* within foods that are "Cereal based products and dishes". The other boxplot to show the distribution of *Protein (g)* within foods that are "Cereal based products and dishes".

Hint: A useful function is boxplot

5 Grouping (2 marks)

Consider the *Food Category* attribute created in Question 3. Draw a bar plot showing *Food category* (x-axis) versus mean *Total sugars (g)* (y-axis).

Hint: Useful functions are groupby and plot (kind='bar')

6 Scatter plot (2.5 marks)

a) Create an attribute *EnergyLevel* which has value "1" if *Energy, with dietary fibre (kJ)* is greater than 1000 kJ and "0" otherwise.

b) Draw a scatter plot showing *Total sugars (g)* (x-axis) versus *Protein (g)* (y-axis) for all foods. A food should have red color if it has "High" energy (*EnergyLevel* of '1') and *blue* if it has "Low" energy (*EnergyLevel* of '0'). There should be an accompanying legend mapping colours to EnergyLevel.

Hint: Useful functions are conditional and scatter plot

7 Parallel co-ordinates (4 marks)

a) For each of the attributes *Protein (g)*, *Total fat(g)* and *Total Sugars (g)*, normalise its values to lie within the range $[0, 1]$ (0 to 1 inclusive). Use the following formula for normalising an attribute:

$$newvalue = \frac{oldvalue - min}{max - min}$$

where *min* is the minimum value for the attribute, *max* is the maximum value for the attribute, *newvalue* is the normalised value for the attribute and *oldvalue* is the old (unnormalised value).

b) Using these normalised attributes, draw a parallel co-ordinates plot, having 1 line for each food. The ordering of the attributes for the plot should be *Total Sugars (g)* first (leftmost), followed by *Total fat (g)*, followed by *Protein (g)*. Colour a food in blue if it is "Low" energy and colour a food in red if it is "High Energy" (using the definitions of *EnergyLevel* from question 6a). Provide a legend mapping colour to *EnergyLevel.*

Hint: Useful functions are parallel coordinates

8 Calorie Counts and Pie Chart (3 marks)

a) Create a new attribute called *calorie_count_per_100g*, which can be computed for each food as follows

$$\begin{aligned} calorie\_count\_per\_100g &= 4 \times (Protein\ (g)) + 4 \times (Available\ carbohydrates,\ with\ sugar\ alcohols\ (g)) + \\ &\quad 9 \times (Total\ fat\ (g)) + 7 \times (Alcohol\ (g)) \end{aligned}$$

b) Print out the top 5 foods with the the highest *calorie_count_per_100g*. The format should be:

```
***
Q8

1. # @
2. # @
3. # @
4. # @
5. # @
***
```

where # is the name of the food and @ is its calorie count per 100 gram, rounded to 1 decimal place.

c) Create a pie chart showing the mean *calorie_count_per_100g* for each *Food category* (from question 3). Each of the 24 slices of the pie should have a different colour and contain a percentage number listing its relative size. Each slice of the pie should have a label next to it indicating which *Food category* it corresponds to.

Hint: Useful functions are pie charts and group by

9 Merge and export to JSON (1 mark) (harder question, leave till last)

a) Create a new attribute *Food category name*, which provides the corresponding name of each (two digit numeric) *Food category*, the attribute that you created in Question 3. You must write code that extracts *Food category name* from data in the provided file "8i. Australian Health Survey Classification System.csv". In that csv file, i) *Food Group Code* can provide information about *Food category*, ii) *Food Group and Sub-Group Name* can provide information for *Food category name*. Note that you will only need food category names that correspond to two digit food categories.

b) Compute the mean *Total sugars (g)* for each *Food category name*. Output this information in JSON format, as follows

```
{
"Mean total sugars (g) by category": {"#": @,
      "#": @,
         ...
      "#": @}
}
```

where each # is a *food category name* and @ is its corresponding mean *Total sugars (g)*, rounded to 1 decimal place.

Hint: Load data from the file "8i. Australian Health Survey Classification System.csv" into a separate dataframe and join this information to the relevant attribute in your food dataframe.

Marking scheme

Correctness (19 marks): For each of the questions, a mark will be allocated for level of correctness (does it provide the right answer, is the logic right), according to the number in parentheses next to each question. Note that your code should work for any data input formatted in the same way as *food_nutrient_2011_13_AHS.csv:*. E.g. If a random sample of 1000 records was taken from *food_nutrient_2011_13_AHS.csv:*, your code should provide a correct answer if this was instead used as the input.

Correctness will also take into account the readability and labelling provided for any plots and figures (plots should include title of the plot, labels/scale on axes, names of axes, and legends for colours symbols where appropriate).

Coding style (1 mark): A Mark will be allocated for coding style. In particular the following aspects will be considered:

- Formatting of code (e.g. use of indentation and overall readability for a human)
- Code modularity and flexibility. Use of functions or loops where appropriate, to avoid highly redundant or excessively verbose definitions of code.
- Use of Python library functions (you should avoid reinventing logic if a library function can be used instead)
- Code commenting and clarity of logic. You should provide comments about the logic of your code for each question, so that it can be easily understood by the marker.

Resources

The following are some useful resources, for refreshing your knowledge of Python, and for learning about functionality of pandas.

- Python tutorial
- Python beginner reference
- pandas 10min tutorial
- Official pandas documentation
- Official mathplotlib tutorials
- Python pandas Tutorial by Tutorialspoint
- pandas: A Complete Introduction by Learn Data Sci
- pandas quick reference sheet
- Python Data Analytics by Fabio Nelli (available via University of Melbourne sign on)

Submission Instructions

Via the LMS, submit a jupyter notebook (A template notebook "notebook-for-answers.ipynb" is provided in the folder with the datasets) containing your Python 3 code.

Other

Extensions and Late Submission Penalties: If requesting an extension due to illness, please submit a medical certificate to the lecturer. If there are any other exceptional circumstances, please contact the lecturer with plenty of notice. Late submissions without an approved extension will attract the following penalties

- $0 < hourslate <= 24$ (2 marks deduction)
- $24 < hourslate <= 48$ (4 marks deduction)
- $48 < hourslate <= 72$: (6 marks deduction)
- $72 < hourslate <= 96$: (8 marks deduction)
- $96 < hourslate <= 120$: (10 marks deduction)
- $120 < hourslate <= 144$: (12 marks deduction)
- $144 < hourslate$: (20 marks deduction)

where *hourslate* is the elapsed time in hours (or fractions of hours).

This project is expected to require 20-25 hours work.

Academic Honesty

You are expected to follow the academic honesty guidelines on the University website `https://academichonesty.unimelb.edu.au`

Further Information

A project discussion forum has also been created on the subject LMS. Please use this in the first instance if you have questions, since it will allow discussion and responses to be seen by everyone. There will also be a list of frequently asked questions on the project page.