COMP20008 Project Phase 1

V1.0: 19th March 2018

Due Date

- Phase 1 (37 marks, worth 17% of subject grade): due 11:59pm 9th April. Submission is via the LMS.

Phase 1: Warmup - Python Exercises

In this phase, you will practice your Python wrangling skills with a publicly available dataset. The dataset includes information about New York City Yellow Taxi trips[1]. The trips include features recording pick-up and drop-off date/time, trip distance, payment amount, duration and passenger count. Table 1 shows the description of each feature.

Field Name	Description
pickup_datetime	The date and time when the taxi meter was engaged
dropoff_datetime	The date and time when the taxi meter was disengaged
trip_distance	The elapsed trip distance in miles reported by the taximeter
passenger_count	The number of passengers in the vehicle (a driver-entered value)
payment_amount	The total amount charged to passengers. Does not include cash tips
duration	The duration of the trip in minutes

Table 1: Summary of Features

You will be working with the following dataset in this phase:

- *clean-january.csv:* a set of trip records (approx 89k) in January 2016. **Note that this dataset is quite large, and you may find it beneficial during development, to first test your code on a smaller sample of this data.**

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 workshop weeks 3-5. If you are using other packages, you must provide an explanation in your code about why it is necessary.

[1] https://data.cityofnewyork.us/Transportation/2016-Yellow-Taxi-Trip-Data/k67s-dv2t

1 Adding a new column to the table (3 marks)

Read the datafile into a pandas dataframe. Using the values in the *trip_distance* column, determine for each trip whether it should be categorised as a short trip. A short trip is one that has a distance of less than 3 miles. Record this information in a new column called *short_trip* where this column has value 1 if a trip is short and value 0 otherwise. Create a new DataFrame with the following schema.

Index	Short_Trip	Trip_distance	Passenger_count	Payment_amount	Duration

and use it to then print out the percentage of short trips. The output of this step should look like

```
Question 1
% of short trips=#
***
```

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

2 Basic statistics (2 marks)

What is the median trip distance and what is the median trip duration? Your code should print out the results with the following format:

```
Question 2 median trip distance
January: #
***
Question 2 median trip duration
January: #
***
```

where STAT is the title of the statistic that you are reporting and # is its value rounded to 1 decimal place.

3 Data cleaning (2 marks)

Consider the feature *trip_distance*. Suppose the 'normal' range for this feature's values is $1.2 \leq trip\_distance \leq 4.5$. Compute the percentage of values that do not fall within this normal range.

The output of this step should be a message printed with the following format:

```
Question 3
#% of instances in 'tripdistance' are abnormal! (normal range is between 1.2 and 4.5)
***
```

where # is the computed value rounded to one decimal place.

4 Visualisation using boxplots (2 marks)

Draw a plot consisting of two boxplots. One boxplot to show the distribution of trip fares during the morning period (trip started during the period 07:00 $\leq$ *time* $\leq$ 11:00). One boxplot to show the distribution of trip fares during the afternoon period (trip started during the period 12:00 $\leq$ *time* $\leq$ 15:00).

tip: pandas.DataFrame.boxplot() draws a boxplot for each column of the DataFrame object
tip: pandas.to_datetime() converts a string to a datetime object. The time and date information are formatted as '%d/%m/%y %H:%M'.

5 Summary (2 marks)

Create a new column *isPeak* that indicates whether a trip was both started (i.e. the meter was engaged) and completed (i.e. the meter was disengaged) in peak hour (07:00 $\leq$ time $\leq$ 08:59 on weekdays). It should have value 1 for peak hour trips and value 0 otherwise. Calculate and print out the percentage of trips that are peak hour trips. The output of this step should look like

```
Question 5
% of peak hour trips=#
***
```

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

6 Grouping (3 marks)

Create a new column called *dayofmonth* which has values in the range 1-31 and records on which day of the month each trip began (i.e. the meter was engaged). Now draw a bar plot showing *dayofmonth* (x-axis) versus total number of trips beginning on that day (y-axis).

7 Scatter plot (4 marks)

Create a column *dayofweek* which records the day of the week (Sunday to Saturday) on which the trip began. Then, compute the mean *trip_distance* and mean *payment_amount* for each day of the week (calculated over all trips which began on that day). Draw a scatter plot showing mean *trip_distance* (y axis) versus mean *payment_amount* for the 7 days of the week. Each day of the week should be plotted in a different colour and there should be an accompanying legend mapping colours to days of the week.

8 Parallel co-ordinates (4 marks)

For each of the features *trip_distance*, *passenger_count* and *payment_amount*, normalise its values to lie within the range $[0, 1]$ (0 to 1 inclusive). Use the the following formula for normalising a feature:

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

where *min* is the minimum value for the feature, *max* is the maximum value for the feature, *newvalue* is the normalised value for the feature and *oldvalue* is the old (un-normalised value).

Using these normalised features, compute the mean *trip_distance*, mean *passenger_count* and mean *payment_amount* for each day of the week. Then, draw a parallel co-ordinates plot, having 7 lines, each corresponding to a different day of the week. The ordering of the features for the plot should be *mean_trip_distance* first (leftmost), followed by *mean_passenger_count*, followed by *mean_payment_amount*. Colour the weekdays in red and the weekends in blue. Provide a legend mapping colours to day type.

9 Pie Chart (4 marks)

Create a new column called *minutes_travelling*, which for each trip is equal to the *passenger_count* multiplied by the *trip_duration*. Create a pie chart showing the mean *minutes_travelling* for each day of the week. Each of the 7 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 day of the week it corresponds to.

10 Another Scatter Plot - harder (5 marks)

For each day of the month (1-31), compute the maximum number of trips occurring within any 60 minute period on that day. This 60min period must start and end on that day, but does not need to begin exactly on the boundary of an hour. It can be defined by the time window Start time $\leq time <$ Start time + 60. Call this *max_trips_60min_day*. Create a scatter plot of 31 points, whose x-axis is day of the month (1-31) and y-axis is *max_trips_60min_day*. Colour weekdays red and weekends blue. Include a legend that explains the colour mapping.

Marking scheme

Correctness (31 marks): For each of the 10 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 *clean-january.csv*. E.g. If a random sample of 10,000 records was taken from *clean-january.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 where appropriate).

Coding style (6 marks): Marks 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 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 Phase 1 assignment for the subject in 2017 was similar to this assignment. On the LMS, the materials used for that assignment, plus a sample answer are provided. You may adapt and reuse these if you find them useful. Note that the sample answer contains minimal code commenting. For your solutions, you should provide a greater degree of comments in your code, typically 6-8 lines for each question.

Submission Instructions

Via the LMS, submit a jupyter notebook (An empty notebook "notebook-for-answers.ipynb" is provided in the folder with the datasets) containing the 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 a penalty of 10% of the marks available per 24hr period (or part thereof) that it is late. E.g. A late submission will be penalised 1.5 marks if 4 hours late, 3 marks if 28 hours late, 4.5 marks if 50 hours late, etc.

Phase 1 is expected to require 18-22 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. The Phase 1 project page will also contain a list of frequently asked questions.