COMP60711 - DATA ENGINEERING
Coursework Description
WEEK 3

Coursework Description

1. Data Description

You have been given access to two road-traffic data files in CSV format, **'rawpvr_2018-02-01_28d_1083 TueFri.csv'** and '**rawpvr_2018-02-01_28d_1415 TueFri.csv**'. Each file contains observations collected via Inductive Loops sensors planted on a particular site of Chester Road in the city of Manchester. For example, file **'rawpvr_2018-02-01_28d_1083 TueFri.csv'** contains observations collected from site 1083, while file '**rawpvr_2018-02-01_28d_1415 TueFri.csv**' contains observations collected from site 1415. The observations were collected during the month of February 2018, one of the busiest winter months of the year. Note that each row of the file contains one observation, i.e., the properties associated with one detected vehicle. As a consequence, if you count the total number of records with the following timestamp (06/02/2018), you are able to estimate the total volume of traffic (i.e., the total vehicle count) on the 6th of February 2018.

Both data files present the same structure, composed of the following attributes or properties:

- ***Date*** is a timestamp containing day of the month and year and time of the day when a vehicle was detected, with the following format: dd/mm/yyyy HH:MM:SS.
- ***Lane*** is an identifier of a given lane of the road (a road may have multiple lanes and each lane has a unique identifier).
- ***Lane Name*** is the name given to a particular lane of the road. Each lane has a unique name.
- ***Direction*** identifies the direction followed by a road lane (e.g., North, South, etc.). Different lanes may follow the same direction.
- ***Direction Name*** is the name of the direction followed by a lane of the road.
- ***Speed (mph)*** is the speed with which the detected vehicle was moving at the time it was detected.
- ***Headway (s)*** is the time distance between two consecutive vehicles following the same route. More precisely, it is the time distance between the front bumper of one vehicle and the front bumper of the vehicle behind it.
- ***Gap (s)*** is also a time distance between two consecutive vehicles following the same route, but it indicates the time distance between the rear bumper of one vehicle and the front bumper of the vehicle behind it.
- ***Flags*** is a number that identifies the day of the week when a vehicle was detected.
- ***Flag Text*** is the text description of the day of the week when a vehicle was detected.

2. Developing the Coursework

The coursework is composed of a list of tasks, divided into three sub-lists. This is the third one, described as follows:

Sub-list 3: This is the third sub-list of tasks and contains three tasks. This sub-list of tasks carries 100 marks and the deadline for its submission is the day of the next timetabled lab session of the following week (Week 4).

When developing the coursework, you should do the following:

1. Use file '**rawpvr_2018-02-01_28d_1083 TueFri.csv**' for all tasks, unless you are explicitly told in the task to use both files, **'rawpvr_2018-02-01_28d_1083 TueFri.csv'** and '**rawpvr_2018-02-01_28d_1415 TueFri.csv**'.

2. When asked to use a technology of your choice to develop your work, use the one you are most familiar with from the suggested ones in Section 4, to avoid steep learning curves.

3. You should upload all your programming solutions into your **Gitlab COMP60711_Part_1 Project**, and answer the essay questions (which require text answers from you) in the **Blackboard Test** associated with this piece of coursework, titled "*60711-Lab2-S-CW2*".

4. To be able to access your Gitlab COMP60711_Part_1 Project, do the following so that an account is automatically created for you:
    a. **Log in to Gitlab using your University username and password.**

5. In more detail, make sure you upload to your Gitlab COMP60711_Part_1 project ALL the Python code you developed for the tasks that require programming, described in Section 3, naming your file as follows: '*CW2.py'.* On Gitlab, you will find the file template with a code skeleton into which you should insert your code, as described in the provided instructions.

    Also, provide explanations of your Python code in the form of comments that you place within the code itself. You can add a paragraph of comment on top of one line or a group of lines of code that you deem to be associated (by functionality).

    Identify and justify any data preparation steps.

    For each comment, use a maximum of *50 words* of text. If the code plots a graph, then make sure that the plotted graph is added to your '*Figures_SolutionCourseworkTasks5to7'* file (see item 6), and that it is referred to in the code comments, using the correct number (see item 6).

6. You should export and paste into a single PDF document any relevant graphs/plots you generate for the tasks described in Section 3 of this document, giving to each graph/plot a distinct number, so that you are able to refer to an individual plot/graph from the text answer you will be submitting via the Blackboard Test titled "*60711-Lab2-S-CW2*". Finally, upload this PDF document, with the graphs/plots, into your Gitlab repository, naming it '*Figures_SolutionCourseworkTasks5to7'.*

7. Note that all your essay-like, text answers (for the tasks described in Section 3) must be written in and submitted via the Blackboard Test titled "*60711-Lab2-S-CW2*", accessible through the course unit's Blackboard space, in folder 'Coursework Material for PART 1 of the Course (Weeks 1, 2 and 3)'. Despite the answers being written on Blackboard, you can refer to any plots/graphs you generated for the tasks in your answers, using the plot/graph number you have specified in file '*Figures_SolutionCourseworkTasks5to7'.*

8. To access the text field areas where your answers should be written, start Test "*60711-Lab2-S-CW2*". Once you have started the Test, you will see all questions relevant to this coursework.

   You must **manually save all your answers** all the time and especially before leaving the test and closing the relevant window browser, in order to be able to resume your coursework at a later time, until the deadline. Use a maximum of 800 words for each task.

   Also, to provide an answer to each of the questions in these tasks, please, START the test. You can interrupt your work in this test at any time to continue at a later time, provided that you MANUALLY SAVE the current draft of your answers without submitting it. If you submit it and begin the test again, you will not see again your previously attempted answers and will need to start from scratch!
   Therefore, you must submit your test only once.
   Once you have finished the test and checked your answers, then you can manually SUBMIT your answers.
   Make sure you SUBMIT your answers INSIDE the coursework DEADLINE.

9. **VERY IMPORTANT:** Note that part of the outcome of Tasks 5.I and 6.I involves numeric value(s) and, so, in the Python code you are going to develop for each of these tasks, you MUST use the Python structures provided in file *'CW2.py',* found from your Gitlab repo, to hold the numeric results. This is necessary to allow the marking of your code to be performed with success.

   When downloading file *'CW2.py'* for running and testing your code, please, make sure you correct the following typo that one of our assistants added to the file:

   - ✓ Replace file name "*rawpvr_2018-02-01_28d_1083 TueFri.csv*" with file name "*rawpvr_2018-02-01_28d_1083_TueFri.csv*"

   - ✓ Replace file name "*rawpvr_2018-02-01_28d_1415 TueFri.csv*" with file name "*rawpvr_2018-02-01_28d_1415_TueFri.csv*"

10. After uploading your *'CW2.py'* file to Gitlab, as well as your '*Figures_SolutionCourseworkTasks5to7"* and '*SolutionTask_7-I'* files, go to our course unit's site on Blackboard and click on the link titled "**COMP60711 Data Engineering Coursework Submission**" to submit your Gitlab project to Blackboard. Upon clicking on the link, through the provided interface, you should press the GitLab

button on the Gradescope submission page (an application that the University uses to assess Gitlab projects) and select the repo to submit the work.

11. Do not ***arbitrarily*** discard/delete rows in the data files provided to you, unless explicitly required in the task question.

3. Tasks Description

Make sure you carefully read the instructions provided in Section 2, before starting developing the tasks described below.

<table>
<tr><th colspan="2">Week3</th></tr>
<tr><td>Task5</td><td>One of the most objective and easy-to-measure data quality dimensions is Completeness. There are different types of Completeness. When dealing with data in tabular format, Column and Row Completeness become relevant.

In this task, we are going to focus on Column Completeness, which is one of the simplest types of Completeness that can be objectively measured.

The Column Completeness function we are going to use is the following:

Column_Completeness = (number_of_non-empty_cells x 100) / number_of_cells

Where the count of the number of non-empty cells in a column and the count of the total number of cells in that column are parameters of this function.

I. Applying the Column Completeness formula above and using Python, measure the level of completeness of the 'Gap (s)' column considering only Tuesdays between 7:00 and 18:59:59 o’clock (i.e., the working hours of Tuesday including morning the evening rush hours).
[3 marks]

II. To improve the quality of the data file, we should seek to fill in the missing values of column ‘Gap (s)’. For simplicity, let us focus on the NB_MID lane (North direction) only, while considering any Tuesday between 7:00 and 18:59:59 o’clock for which the value for this column is missing.

But, before choosing a method to fill in the missing values of column ‘Gap (s)’, using Python, build a profile of the column’s data values to find out its relevant details. This profile of the values present in the column should help you decide how the missing values should be filled, by providing you with an idea of what data values could possibly represent as best as possible the “real” values that are missing, given the present data and use case at hand.

Following the profiling exercise, enumerate the profiling techniques you used and what each one has allowed you to find out about the data, using the relevant text field (the one associated with this task) in the relevant Blackboard Test (the one associated with this coursework).

Also, following the instructions provided in Section 2, upload to Gitlab any commented/explained Python code and any associated plotted graphs you</td></tr>
</table>

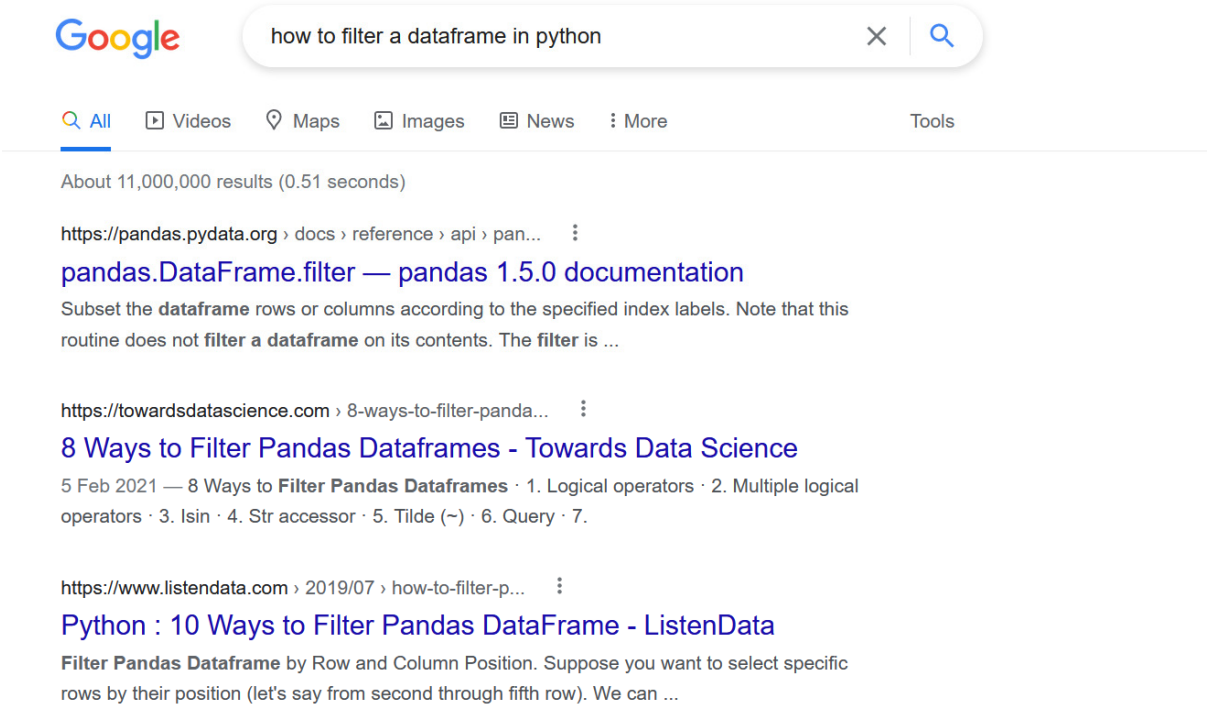
		generated for this task, describing method application and your decision-making. (*HINT: try to obtain a detailed profile of the column (rather than a minimal one).* **[6 marks]** III. Considering the variety of ways in which a numeric column, such as 'Gap (s)', can have its empty cells filled with values, choose a method to fill in the missing values for this particular column (<u>from the suggested below</u>), having as a basis the profile you generated in Task 5.II, and considering the use case at hand. For facilitating the marking of this task, choose one of the following methods: 1) replacing all missing values with a single constant; or 2) replacing all missing values with the AVG of all the column values; or 3) replacing each missing value with the AVG of the column values of a given group of records; or 4) replacing all missing values with the median value of the column; Write in the relevant answer text field (the one associated with this task) of the relevant Blackboard Test (the one associated with this coursework) the method you chose (from the list provided above) to fill in the missing values of column 'Gap (s)', making sure your choice of method is clear, by ***explicitly*** specifying what it is, and making sure your choice is justified. Additionally, suggest one method, <u>outside of the above list</u>, which is (also? or more?) suitable according to the profile of this particular column. Justify your choice using the profile you generated in Task 5.II as a basis, and considering the use case at hand. Write it all on the relevant Blackboard Test. Following the instructions provided in Section 2, upload to Gitlab any commented/explained Python code and associated plotted graphs you generated for this task, describing method application and your decision-making. **[8 marks]**
Task6	I.	To further polish your analytical skills, using Python, estimate the typical Friday Journey Time (JT) for the road fragment between site 1083 and site 1415, between 17:00 and 17:59:59, using only the North direction lanes. JT is a very popular traffic application among road users, via which the length of time period that it takes to travel through a road fragment is estimated. You can use the very simple formula for estimating JT, described as follows: **JT = distance/speed** Assume that the distance between the two points in these two sites is 4.86km. Note the units of distance and time and consider when conversion between units is necessary. Also, use the average speed, considering the relevant time period (17:00 and 17:59:59) and lanes. To calculate the average speed between 17:00 and 17:59:59 of all North lanes, consider not only the three North lanes of site 1083, but also the two North lanes of site 1415. Make sure your JT is given in minutes, which is what road users normally get from popular applications such as Google Maps. Write as, an answer to this question, any data quality checks/profiling

		and/or data preparation you had to do prior to calculating JT and any challenges you encountered when doing the calculation. Following the instructions provided in Section 2, upload your commented Python code and any plotted graphs to Gitlab. **[8 marks]**
Task7	I.	Choose a technology other than Python (from the suggested in Section 4) and repeat *Task_6.I.* Make sure you paste/export the step-by-step code/recipe you developed using the technology of your choice. As detailed in Section 2, upload any code or exported recipe representation to Gitlab, giving the relevant file the name '*SolutionTask_7-I'.* Also, describe the features of the technology, emphasising the ones that have made your work easier or more difficult, in contrast to Python. In your description, you can make comments about functionality that is not offered within the technology, rendering you unable to satisfactorily complete *Task 6.I*. Write this in the relevant answer text field of the relevant Blackboard Test, as described in Section 2. **[6 marks]**

4. Further Advice

The following list contains the PL and Data Manipulation/Preparation tools we suggest you use in the development of your coursework: *Python, Knime, OpenRefine and MySQL*. These are of easy access and free installation. You can access them from the machines in lab, or you can just download and install them in your machine.

If you are not familiar with the suggested programming language and/or tools, then you can search for commands in the Web, as shown below:



If using Python, for example, you will be interested in using commands from packages such as *pandas*, *numpy*, *datetime*, *os* and *calendar*, to handle *Date* related data types.

General tutorials for each of these can be found from the following links:

- **Knime** (https://www.knime.com/downloads/download-knime)
  Documentation: (https://docs.knime.com/)
  Tutorials: (https://www.youtube.com/watch?v=HEp9Cbql2hs,
  https://www.youtube.com/watch?v=5WAyOiIfHPg)

- **OpenRefine** (https://openrefine.org/download.html)
  (https://www.youtube.com/watch?v=WCRexQXYFrI),
  (https://www.youtube.com/watch?v=wGVtycv3SS0),
  (https://www.youtube.com/watch?v=wfS1qTKFQoI)

- **Python** (https://www.tutorialspoint.com/python/index.htm)

- **MySQL** (https://www.mysql.com/downloads/)
  (https://www.tutorialspoint.com/mysql/index.htm)