INFOH600 Computing Foundations of Data Science

Project Assignment

2019-2020

Introduction

Data Science is all about analyzing data. Before data is suitable for analysis, however, it often needs to be *wrangled*, which means that it needs to transformed into a shape that is suitable for analysis. In particular, data often needs to be integrated, cleaned, and transformed, before any analysis can be made.

The goal in this project is to use the tools introduced in INFOH600 (namely: Python, its data science libraries, and big data tools) to do a conceptually relatively simple analysis of a real-world dataset. Data wrangling is an essential part of the project.

1 The Data

The New York City Taxi and Limousine Commission (or TLC for short) has been publishing records about taxi trips in New York since 2009. The complete dataset, as well as documents that describe these data, is available on the following website: `https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page`

The TLC trip dataset actually consists of 4 sub-datasets:

- Yellow taxi records are records that record trip information of New York's famous yellow taxi cars.
- Green taxi records are records that record trip information by so-called 'boro' taxis — a newer service introduced in August of 2013 to improve taxi service and availability in the boroughs.
- FHV records (short for 'For Hire Vehicles') record information from services that offer for-hire vehicles (such as Uber, Lyft, Via, and Juno), but also luxury limousine bases.
- High volume FHV (FHVHV for short) are FHV records offered by services that make more than 10,000 trips per day.

For a complete description of these sub-dataset types, see `https://www1.nyc.gov/assets/tlc/downloads/pdf/trip_record_user_guide.pdf` and the links therein.

Note that not all sub-datasets provide the same information. Again, see the above-mentioned URL and the data dictionaries linked therein for a complete description.

Each sub-dataset consists of a set of files. Each such file records the trip information within a given month (for the given subset type). For example, `yellow_tripdata_2009-01.csv` contains all Yellow Taxi records produced in january 2009. All files are in CSV (Comma Separated Values) format.

Because the full TLC dataset is rather large (more than 280 GB at the time of writing), you are asked to do this assignment on a 5% fraction of it, which can be downloaded from the following URL:

`https://davinci.ulb.ac.be/index.php/s/Grtw6L3RJgbaZzo`

(This compressed file is 2.9 GB large, yielding 14 GB uncompressed.)

The files in this download in one-to-one correspondence to the original dataset. For example, `yellow_tripdata_2009-01.csv` in the download was obtained by uniformly sampling the corresponding file `yellow_tripdata_2009-01.csv` from the TCL website (without repetition) until 5% of the records were retrieved.

2 Assignment

You are asked to complete the following analysis tasks over the sampled TLC dataset.

2.1 Collecting metadata, inspecting schema evolution

Any analysis in Data Science starts with understanding the data. Hence, your goal in this first task is to get an understanding of the data and its characteristics. To this end, first read the description(s) on the above-mentioned websites.

Next, answer the following questions in your report, for each of the different sub-dataset types (Yellow Taxi, Green, FHV, FHVHV) in the sample.

- Compute basic statistics about the number of files in this sub-dataset, their size, and the number of records (lines) in each file. For length and number of records, give the min, max, mean, 25, 50, 75, 90 percentiles values.
- The first line of each file gives the *schema* of the data in the file: that is, it gives a name to each column that indicates the kind of data in that column.[1] Unfortunately, this schema has changed a number of times over the past decade. For example, the order of the columns has sometimes changed, some columns have been renamed over time, new columns have been added that were not available before, and other columns that were available have been dropped.

  Analyze these schema changes:

  - Which files in the sub-dataset have the same schema? Posed differently: from what start date (specified by year + month) until what end date was a specific schema version in use?
  - Summarize the changes from one schema version to the next: how have columns been renamed? Which columns are new ? Which columns are removed ? Hypothesize whether some of the new columns can be computed from old columns. Was the order of the columns changed?

[1]These names and their contents are also described in the data dictionaries available on the TLC website.

Note: you will find that often the same column name is capitalized differently from one file to the next. For example, one file may have `tpep_pickup_datetime` as a column name, while the next has `Tpep_Pickup_Datetime`. To keep changes manageable, we will not consider this a renaming: two column names are considered the same if they are identical after converting them both to lower case letters. By contrast, if a column was first named `pickup_datetime` and later `tpep_pickup_datetime`, then we would consider this a renaming.

Note that the number of files in a sub-dataset can be large. It is hence *not* the idea that you manually start browsing the schema of each file. Instead, write code to extract the schema, detect that two schemas are the same, compute the difference between two schemas, and divide the dataset into classes of files that use the same schema. Once that is done, you can manually start inspecting the schema differences that you have computed between versions.

2.2 Data integration

The goal in Section 2.4 below will be to answer certain queries that are formulated over the *last* schema version in each sub-dataset. Nevertheless, we would like to use all data in the sub-dataset when answering the queries. In order to make this possible, you must first *integrate* these sub-datasets. Specifically, for each sub-dataset, write (and execute) code that converts a file (using possibly an old schema) into a file that has the new, latest schema version.

You logic for doing this should be as follows:

- If a column exists in the old schema version, but not in the new one, drop its contents.
- If a column exists both in the old schema version and the new (but possibly under a different name, or the order of the column in the new schema version is different), copy the value.
- If a column does not exist in the old, but does exist in the new, give it a blank value (indicating that the record does not have such a value).
- If a column does not exists in the old schema, but does exist in the new, and it is possible to compute the value of the new column given information in the old schema, then fill in this computed value.

An example of the last item is the `PULocationID` column in the Yellow taxi dataset. This column specifies a geographic area in which the taxi trip started. Older files have `pickup_longitude` and `pickup_latitude` columns which are much more precise, and from which the 'PULocationID' value can be computed (using the information about taxi zones on the TCL website).

Your conversion code should *not* modify the original files, but instead create a new file.[2]

Explain the design behind your conversion functions in your report.

The data integration step is highly parallellizable. Therefore, your solution on this part *must* be written in Spark.

[2]This has the added benefit that if, during early development your code is somehow incorrect, the original data is still there and you do not need to re-download it.

2.3 Data cleaning

Now that the files in each sub-dataset conform to the same schema it becomes possible to check the files for errors. To this end, for each sub-dataset:

- First analyze what the valid values for each column are, based on the data dictionary available at the TLC website. (E.g., `trip_distance` should be a floating point value, and cannot be negative). Be sure to discuss your conclusions in your report.
- Write code that can be used to check all the validity constraints for this sub-dataset.
- For each file in the sub-dataset, detect the "dirty" records by running your validity constraints.
- Inspect the dirty records. Are there records that can be repaired ? Repairing is possible, for example, when the record has data of the incorrect type in the column, but it is clear how to convert that data to the desired type.
- Split the file into two: one containing the clean and repaired records, and one containing the dirty (and unrepairable) records. During the analysis of Section 2.4, we only use the clean and repaired records, the others are discarded.
- In your report, include a discussion that summarizes, for each file, the kinds of errors found (with statistics), how you repaired those errors (if applicable), and how many records are discarded (because they are dirty and unrepairable).

2.4 Analysis

For each of the following queries, write code to compute the answers. Include a plot of these results in your report (plotted using matplotlib). Briefly discuss what you can observe from these results. Beware that for some queries, not all of the sub-dataset types have the information required to answer the query. In that case, please ignore those sub-datasets.

- The monthly total number of trips, grouped per dataset type. When plotted over time (i.e., with the months in chronological order on the x-axis), this query should allow you to see how the popularity of a particular service (Yellow Taxi, Green Taxi, FHV, FHVHV) is evolving over time.
- Monthly total number of trips in Manhattan and Brooklyn, grouped per dataset type. Plot over time.
- The monthly total receipts, grouped per dataset type. Here, the total receipt is the sum over all receipts in the same month. This should exclude tips, but include fares, surcharges, taxes and tolls. Plot over time. Which service provider is making the most revenue ?
- The average trip receipt, where the average is computed over trips within the same month, grouped per dataset type. Receipts should exclude tips, but include fares, surcharges, taxes and toll. Plot over time. (Note: this query can only be evaluated for those sub-dataset types that include receipt information.)

- The average cost per in-progress-minute, where the average is computed over trips within the same month, grouped per dataset type.[3] Cost should exclude tips, but include fares, surcharges, taxes and tools. Plot over time. Which service is hence cheapest on average ?
- The average tip per trip, where the average is computed over trips within the same month, grouped per dataset type. Cost should exclude tips, but include fares, surcharges, taxes and tools. Which service would you hence prefer to be in, as a driver (and assuming that you don't have to share tips)?
- The median monthly average trip speed, where the median is computer over trips within the same month, grouped per dataset type and per borough[4]. Plot over time.
- How long does it take to get to a New York City Airport? New York City has 3 airport: Newark Airport, JFK, and LaGuardia Airport. Assume that the trip to the airport starts in Manhattan Midtown (zones 161–164). This query can be answered from different perspectives. First, compute the median travel time from Manhattan Midtown to the airport under consideration, binned per departure time (e.g., departure between midnight and 1h, between 1h and 2h, ...). Here the median is computed over all trips. Plot the results, per airport. What is the best timeslot to depart to the airport? What is the worst timeslot? For the best and worst timeslots, also compute how the monthly median travel time has evolved over the past years.

3 Deliverables

You need to deliver both your implementation used to resolve the tasks above, and a report that documents your solution approach (motivating your approach) and answers the questions posed above. It is required that your report also documents your execution environment (e.g., specify the libraries that you have used in your code, as well give the specification of the machine(s) on which your code was executed).

It is strongly preferred that your report takes the form of Jupyter Notebooks, which contains hence both your code and the execution's results, plots, and answers.

A template for the set of solution notebooks, as well as scripts that can be used to download the datasets, is available at the following URL: `https://davinci.ulb.ac.be/index.php/s/ZSyZDB46nLXzeEB`

4 Modalities

The assignment has the following modalities:

1. The project assignment constitutes the full grade on INFOH600. There is no exam. This implies that the project is mandatory. If you do not make the project, you cannot pass the course.

[3]To clarify: assume that within the month under consideration there were only two trips. The first costs 10$ and lasts 2 minutes, the second costs 20$ and lasts 3 minutes. The average cost per in-progress-minute for this month is hence $\frac{10+20}{2+3} = 6\$$.

[4]The NYC boroughs are Manhattan, Brooklyn, Queens, the Bronx, Staten Island.

2. The project will be graded on (1) the implementation itself and (2) the report that you need to write to describe your and motivate your design and implementation.

3. To solve the project, you are allowed to use *only* the tools introduced in INFOH600. I.e., you are allowed to use python and any of its libraries (e.g., Pandas) as well as big data tools such as Map/Reduce and Spark. You are *not*, allowed, however, to resort to other tools to do the analysis. For example, you are *not* allowed to load the data in a relational database, and answer the analysis queries from there. Nor are you allowed to implement the tasks using a different programming language (e.g., Scala, Java, or shell scripts).

4. The project should be solved in groups of 2 (if the total number of students in the course is not divisible by 2, at most one groups of 3 students will be allowed). You are asked to register, per group, the names of the group members via the online poll available at

   `https://docs.google.com/forms/d/e/1FAIpQLSfNVlFhOHiKv9CljUveuUtJFm55ZX7rxHTkhPOiyOZUN5_k-g/viewform`

   by March 6 at the latest. If you cannot find a partner, please indicate so by sending an email to prof. Vansummeren (`svsummer@ulb.ac.be`), who will hook you up with a partner.

5. You will have to create a GIT repository[5] in the `INFO-H-600/2019-2020-s1` repository group at `http://wit-projects.ulb.ac.be/rhodecode/` to submit your report and your code. The username and password to login to this system correspond to your ULB/VUB NetID. The repository will be named

   ```
   project-<student1>-<student2>
   ```

   where `student` corresponds to your student number and `<student1>-<student2>` appear in sorted (alphabetic) order. This repository must be made private. It is recommended that you create this repository *as soon as possible* to avoid last minute technical difficulties, and that you use it throughout the project to synchronize your changes.

   Tip. If you attempt to push a large set of changes to a GIT repository with HTTP or HTTPS, you may get an error message such as `error: RPC failed; result=22, HTTP code = 411`. This is caused by a GIT configuration default which limits certain HTTP operations to 1 megabyte. To change this limit run within your local repository

   ```
   git config http.postBuffer *bytes*
   ```

   where `*bytes*` is the maximum number of bytes permitted.

   Note. Do *not* include the dataset files in your repository, nor your integrated / cleaned datasets! These are simply to large to upload to the GIT repo.

6. Your solution should be pushed to the repository *no later than 10 May 2020.* You get a penalty of -1/6 points for each day that your solution is delayed. Only the latest commit will be considered as the solution.

[5]`http://git-scm.com/documentation`

> *Art.34 In case of fraud or plagiarism during an examination or during a test at an interim date during the academic year, or in relation with the preparation of written reports or papers, the course professor reports the case in writing prior to the jury deliberation to the relevant academic authority levels for disciplinary matters. A copy of that fraud report is addressed to the jury chairmen. The student can ask to be heard by a jury chairperson prior to the jury deliberation, in presence of the related course professor. Without prejudice to the disciplinary processes at the University Faculty level, in case of fraud the student points for the related course are brought down to 0/20. The jury further can:*
>
> - *decide to cancel the examination session;*
> - *decide to refuse the student access to both examination sessions of that academic year.*

Figure 1: Excerpt of the Exam Regulations concerning fraud.

7. Sharing of code or reports between groups is not allowed. (Groups may, however, verbally discuss ideas on how to tackle the project).
8. Plagiarism, in the sense of copy-pasting from existing reports or books is not allowed. To avoid plagiarism, be sure to always quote your sources and indiate clearly if something has been copied verbatim. In case plagiarism is detected, students risk being punished according to article 34 of the exam regulations shown in Figure 1.
9. As stated above, this project assignment is done in groups of 2. All members of the group receive the same grade on the project! Therefore, in case that a group member feels that another group member does not do his/her share of the agreed work, please contact prof. Vansummeren immediately.
10. In case of questions related to the project, please contact prof. Vansummeren (`svsummer@ulb.ac.be`).

5 Some words of advice

- Start working on the project as soon as possible. It may take longer than you initially think to complete.
- The dataset is large enough that computation on it requires time. To save time during development and debugging, consider first solving the tasks using small samples of the entire (already sampled) dataset.
- Be intelligent in how you execute your queries. Some queries are quite similar, and hence it may be possible to first compute common subresults from which the final query answers can be computed efficiently (thereby avoiding the re-computation of these subresults for each query).
- You are allowed (but not required) to use the Big Data cluster that was used during the exercises to solve the project. The raw dataset is already downloaded on the cluster, and accessible at the following path.

`/home/hpda000034/infoh600/sampled/`

and on HDFS at

`hdfs:///user/hpda000034/infoh600/sampled/`

To avoid overloading the cluster, please refrain from re-downloading the dataset in your own home directory.

- When you decide to use the cluster, be mindful that your peers may also use it (and that your jobs may hence execute slowly or that they are put on pause until running jobs complete). This is especially true close to the project deadline. Therefore, starting earlier is better.
- You may be wondering how you can compute a LocationID from a (pickup or dropoff) longitude and latitude. The solution template available at `https://davinci.ulb.ac.be/index.php/s/ZSyZDB46nLXzeEB` has a jupyter notebook that explains how this can be done.