COMP 370 Homework 4 – 311 Data Analysis

Assigned Sept 19, 2024
Due Sept 27, 2024 @ 11:59 PM

Conceptual Questions

1. Thinking about motivation for a data science project, write out a clear (and plausible) use case for the Technical Exercise below (following these Conceptual Questions).
2. Refine the following question into a data science question. Note that for some steps, you'll need to decide on the refinement that creates operational clarity (i.e., you don't have a stakeholder to ask). Show the refinement steps as shown in Lesson 35, Slide 5. Draw on your experiences from class and past homeworks: "We'd like to understand how much characters talk across the My Little Pony series."
3. Explain why state maintenance is hard to do in a Jupyter notebook.
4. Make an argument for why a Jupyter notebook is better than a README for sharing a data science project with data scientists that might engage with it in the future.

Technical Exercise

In this homework, you are a data scientist working with the New York City data division. Your task is to develop a dashboard allowing city leaders to explore the discrepancy in service across zipcodes. You'll be using a derivate of the following dataset:

- https://data.cityofnewyork.us/Social-Services/311-Service-Requests-from-2010-to-Present/erm2-nwe9

NOTE: The original dataset is very large and therefore we have provided a subset of it.

For the purpose of this assignment:

1. Download `nyc_311.csv.tgz` from MyCourses.
2. Trim it down to only include the incidents that occurred in 2020 (for an added challenge, see if you can trim the dataset down using exactly one call to the grep command line tool).

For the remainder of this assignment, you should only work with the trimmed down dataset.

Task 0: Setup a git repo

Setup a repo for this project (or for all your COMP 370 homeworks) on Github (or your favorite git repository host).

Check the repo out to your machine.

Task 1: CLI investigation tool

Build a python-based command line tool borough_complaints.py that uses argparse to provide a UNIX-style command which outputs the number of each complaint type per borough for a given (creation) date range.

The command should take arguments in this way:

> borough_complaints.py -i <the input csv file> -s <start date> -e <end date> [-o <output file>]

If the output argument isn't specified, then just print the results (to stdout).

Results should be printed in a csv format like:

> complaint type, borough, count
> derelict vehicles, Queens, 236
> derelict vehicles, Bronx, 421
> …

Note that borough_complaints.py -h should print a relatively nice help message thanks to argparse.

Commit your script, but not the data to your git repo.

Task 2: Get Jupyter up and running on your EC2

Setup Jupyter on your EC2 as shown in class.

Task 3: Use your CLI tool and jupyter together

In Jupyter, create a bar chart showing the number of occurrences of the overall most abundant complaint type over the first 2 months of 2020. (note: before making the plot, you'll need to figure out the complaint type to use)

How does this compare to that same complaint type in June-July of 2020?

Task 4: Build a Bokeh dashboard

The goal of your dashboard is to allow a city official to evaluate the difference in response time to complaints filed through the 311 service by zipcode. Response time is measured as the amount of time from incident creation to incident closed. Using the dataset from the previous exercise, build a bokeh dashboard which provides in a single column, the following:

- A drop down for selecting zipcode 1
- A drop down for selecting zipcode 2
- A line plot of monthly average incident create-to-closed time (in hours)
    - Don't include incidents that are not yet closed
    - The plot contains three curves:
        - For ALL 2020 data
        - For 2020 data in zipcode 1
        - For 2020 data in zipcode 2
        - A legend naming the three curves
        - Appropriate x and y axis labels

When either of the zipcode dropdowns are changed, the plot should update as appropriate.

Other details:

- Your dashboard should be running on **port 8080**. The **dashboard name (in the route) should be "nyc_dash"**.
- On any change to either zipcode, your dashboard must update within 5 seconds.
- A design tip: there are WAY too many incidents in 2020 for you to be able to load and process quickly (at least quickly enough for your dashboard to meet the 5 second rule). The way to solve this is to pre-process your data (in another script) so that your dashboard code is just loading the monthly response-time averages for each zipcode … not trying to compute the response-time averages when the dashboard updates.

How to solve this assignment - Hints

1. You can trim down a subset of the 2020 dataset and do experiments on your local machine
2. Depending on the EC2 instance type you have, it might not have the processing power you need. In that case, it might be better to do this on your own machine. But it's 100% worth starting on your EC2 and then seeing if you run into an issue.

FAQ

1. For some rows the closed date is before the open date resulting in a negative response time. How to handle those rows? - *These rows should be removed. You can use some filter.*

2. If the start date is in Jan and close date in Feb, which month does it belong to? - *Feb. Since that is when the issue was resolved.*
3. Do rows with missing zip codes be included in the overall average? - *These should be removed. At the beginning when you trim the data, you should only choose the rows with zipcode.*
4. Do we include cases based on "open in 2020" or "closed in 2020"? - *Open in 2020.*
5. Should we calculate duration by day or by hour? - *In hours.*
6. What if there is no end date for a zipcode event? - *Remove it.*

Submission Instructions

- complaint_borough.py – submit your script from Task 1
- complaint_type_analysis.md – explain what your analysis revealed from Task 3
- Bokeh.tgz – the source code (ONLY source code) for your bokeh dashboard from Task 4