DataEng S22: Data Validation Activity

High quality data is crucial for any data project. This week you'll gain experience with validating a real data set.

Submit: Make a copy of this document and use it to record your results. Store a PDF copy of the document in your git repository along with any needed code before submitting using the in-class activity submission form.

Initial Discussion Question - Discuss the following question among your working group members at the beginning of the week and place your responses in this space. Or, if you have no such experience with invalid data then indicate this in the space below.

Have you ever worked with a set of data that included errors? Describe the situation, including how you discovered the errors and what you did about them.

Response 1: I built a data scrapper that took data from multiple e-commerce websites. There were certain data that would not match certain fields when scrapped because of the positioning, I had to look into that and fix them.

Response 2: The data that we used in week2 had errors in them. We had discovered errors in that and then fixed them using regular expressions and python functions.

Response 3: N/A

Response 4: N/A

The data set for this week is a listing of all Oregon automobile crashes on the Mt. Hood Hwy (Highway 26) during 2019. This data is provided by the Oregon Department of Transportation and is part of a larger data set that is often utilized for studies of roads, traffic and safety.

Here is the available documentation for this data: description of columns, Oregon Crash Data Coding Manual

Data validation is usually an iterative three-step process.

A. Create assertions about the data
B. Write code to evaluate your assertions.
C. Run the code, analyze the results and resolve any validation errors

Repeat this ABC loop as many times as needed to fully validate your data.

A. Create Assertions

Access the crash data, review the associated documentation of the data (ignore the data itself for now). Based on the documentation, create English language assertions for various properties of the data. No need to be exhaustive. Develop one or two assertions in each of the following categories during your first iteration through the ABC process.

1. *existence* assertions. Example: “Every crash occurred on a date”
    a. Each crash has a Vehicle ID
    b. Each crash of record type 1 has a Serial #
2. *limit* assertions. Example: “Every crash occurred during year 2019”
    a. Crash Id of length 8
3. *intra-record* assertions. Example: “If a crash record has a latitude coordinate then it should also have a longitude coordinate”
    a. For every crash id there must be a latitude and longitude
    b. For every participant there must be age
    c. For every participant there must be gender
4. Create 2+ *inter-record check* assertions. Example: “Every vehicle listed in the crash data was part of a known crash”
    a. For every crashId there will be atleast one vehicle id
    b. For every crashId, there will be atleast 2 types of records
5. Create 2+ *summary* assertions. Example: “There were thousands of crashes but not millions”
    a. CrashId is unique across all the records
    b. For every crash, the number of participants is always more than or equal to the number of cars
6. Create 2+ *statistical distribution assertions*. Example: “crashes are evenly/uniformly distributed throughout the months of the year.”
    a. Crashes are more during the middle of the day (office hours)
    b. Crash data is evenly distributed throughout the year

These are just examples. You may use these examples, but you should also create new ones of your own.

B. Validate the Assertions

1. Study the data in an editor or browser. Study it carefully, this data set is non-intuitive!.

2. Write python code to read in the test data. You are free to write your code any way you like, but we suggest that you use pandas' methods for reading csv files into a pandas Dataframe.
3. Write python code to validate each of the assertions that you created in part A. The pandas package eases the task of creating data validation code.
4. If needed, update your assertions or create new assertions based on your analysis of the data.

C. Run Your Code and Analyze the Results

In this space, list any assertion violations that you encountered:

- I found around 20 records that did not have age, the value was nan. For this violation I removed the data because I did not have any reference.
- I found 483 records that did not have a proper value for sex, the document said it has to be one of 1,2,3,9. But those 483 records had the value of 4. For this violation I removed the data because I did not have any reference as to what that data meant, the coding manual only referenced 4 types, and this was a weird type.

For each assertion violation, describe how to resolve the violation. Options might include:

- revise assumptions/assertions
- discard the violating row(s)
- Ignore
- add missing values
- Interpolate
- use defaults
- abandon the project because the data has too many problems and is unusable

No need to write code to resolve the violations at this point, you will do that in step E.

D. Learn and Iterate

The process of validating data usually gives us a better understanding of any data set. What have you learned about the data set that you did not know at the beginning of the current ABC iteration?

I learnt that the for majority of the crashes, the number of participants is always more than the number of vehicles. I was also expecting that the number of crashes will be highest during the evening/night time, but in fact, there were equal number of crashes in the morning time as well. And it was spread out throughout the working time slots.

Next, iterate through the process again by going back through steps A, B and C at least one more time.

E. Resolve the Violations and Transform the Data

For each assertion violation write python code to resolve the violation according to your entry in the “how to resolve” section above.

Output the validated/transformed data to new files. There is no need to keep the same, awkward, single file format for the data. Consider outputting three files containing information about (respectively) crashes, vehicles and participants.

Added the code in to Jupyter Notebook and also added the CSV files into the github.