DataEng: Data Validation Activity

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 for this week.

High quality data is crucial for any data project. This week you'll gain some experience and knowledge of analyzing data sets for quality.

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. First (part A) you develop assertions about your data as a way to make your assumptions explicit. Second (part B) you write code to evaluate the assertions and test the assumptions. This helps you to refine your existing assertions (part C) before starting the whole process over again by creating new assertions (part A again).

Submit: In-class Activity Submission Form

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 for this assignment, two or more assertions in each category are enough.

1. Create 2+ *existence* assertions. Example, "Every record has a date field".
    - crashID is unique for every crash and should not be NULL
    - Every record has a date starting from 1/1/2019
    - Every record has a highway number 26

2. Create 2+ *limit* assertions. The values of most numeric fields should fall within a valid range. Example: "the date field should be between 1/1/2019 and 12/31/2019 inclusive"
    - The record type field ranges from 1 to 3
    - All the crash type ranges from 2 to 5
    - The collision type ranges from 0 to 6

3. Create 2+ *intra-record check* assertions.
    - The crash hour is 99 if the hour is unknown and if a crash occurs at 11:01 AM and another at 11:59 AM then both are coded at crash hour 11 and the format is military time.
    - Crash date is a combination of crash day, month and year.
    - The county code with the five character DMV accident number makes up the unique crash ID.
    - When highway number is entered then the Impact Locationbe a numeric value <= 14
    - Every crash ID should be 7 digits long.
    - All the vehicle ID associated with crash ID should not be greater than 7 digits long.

4. Create 2+ *inter-record check* assertions.
    - ALCHL_INVLV_FLG. Every crash has a alcohol involved flag which takes values 0 or 1 indicating if the crash involved the participant consuming alcohol.
    - Crash severity is code number is 1, 70% of the time
    - 68% of the time the weather when the accident occured is ‘Clear’
5. Create 2+ *summary* assertions. Example: “every crash has a unique ID”
    - Every crash has a unique participant ID
    - Every crash has a unique vehicle ID.
    - Every crash has a unique vehicle coded sequence number.

6. Create 2+ referential integrity insertions. Example “every crash participant has a Crash ID of a known crash”

    - City ID is null for all those accidents that occurred outside city limits.
    - NHS flag takes value 0 or 1 based on whether a crash occurred on a national highway or not
    - Vehicle ID is generated for every crash that has occured .

7. Create 2+ *statistical distribution assertions*. Example: “crashes are evenly/uniformly distributed throughout the year.”
    - Crashes are distributed over the different days of a week
    - The majority of the highway system is regular mileage i.e mileage type is 0

B. Validate the Assertions

1. Now study the data in an editor or browser. If you are anything like me you will be surprised with what you find. The Oregon DOT made a mess with their data!
2. Write python code to read in the test data and parse it into python data structures. You can 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. Again, pandas makes it easy to create and execute assertion validation code.
4. If you are like me you'll find that some of your assertions don't make sense once you actually understand the structure of the data. So go back and change your assertions if needed to make them sensible.
5. Run your code and note any assertion violations. List the violations here.

- The vehicle ID should be unique for every crash ID but, I find that there are few repetivie vehicle IDs for the crash IDs.
- The age of participants in the crash must be categorized as 00,01 and 99 when their age is not known, <= two years and >= 98 respectively but the sheet has ages varying from 0 to 9
- The speed involved flag in the sheet should have been a yes or a no based on exceeding the posted speed but it takes values US and OR in the sheet instead.

C. Evaluate the Violations

For any assertion violations found in part B, describe how you might resolve the violation. Options might include "revise assumptions/assertions", "discard the violating row(s)", "ignore", "add missing values", "interpolate", "use defaults", etc.

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

If you chose to "revise assumptions/assertions" for any of the violations, then briefly explain how you would revise your assertions based on what you learned.

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 ABCD iteration?

Next, iterate through the process again by going back to Step A. Add more assertions in each of the categories before moving to steps B and C again. Go through the full loop twice before moving to step E.

E. Resolve the Violations

For each assertion violation found during the two loops of the process, write python code to resolve the assertions. This might include dropping rows, dropping columns, adding default values, modifying values or other operations depending on the nature of the violation.

Note that I realize that this data set is somewhat awkward and that it might be best to “resolve the violations” by restructuring the data into proper tables. However, for this week, I ask that you keep the data in its current overall structure. Later (next week) we will have a chance to separate vehicle data and participant data properly.

E. Retest

After modifying the dataset/stream to resolve the assertion violations you should have produced a new set of data. Run this data through your validation code (Step B) to make sure that it validates cleanly.

Submit: In-class Activity Submission Form