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".**
    - Every record should have carsh ID, vehicle ID
    - Unique serial number.
    - Every record should have highway 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"**
    - 41 to 46 Latitude Degrees

- 0 to 59 Latitude Minutes
- 0.00 to 59.99 Latitude Seconds
- A crash hour should be between 0-24.
- If entered, city value>0
- Traffic control device should have values either 0 or 1

3. **Create 2+ *intra-record check* assertions.**
    - Distance from Intersection must be > 0 when Road Character is not 1 (Intersection) and Milepoint is not provided. Distance from intersection must =0 when the road is 1.
    -
4. **Create 2+ *inter-record check* assertions.**
    - 99% of crashes have injured either pedestrians or non-motorists and not the people who have used safety equipment use.
5. **Create 2+ *summary* assertions. Example: "every crash has a unique ID" Every county ID**
    - The crash ID should be unique.
    - Vehicle ID should be 7 digits.
    - If the LRS field is blank for crashes that don't occur on the state highway system.

6. **Create 2+ referential integrity insertions. Example "every crash participant has a Crash ID of a known crash"**
    - County code is not blank and special jurisdiction code must be cross referential.
    - Every record should have a participant ID for the corresponding crash ID and vehicle ID.

7. **Create 2+ *statistical distribution assertions*. Example: "crashes are evenly/uniformly distributed throughout the year."**
    - On average, more crashes occur when the weather is clear.
    - All the crashes occurred during the day light.

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.**
   Violation 1: The crash hour is not in between 0-24 hours. It has one invalid value called 99.0
   Violation 2: Vehicle ID value is of length 7 except for entries whose value is 0.0
   Violation 3: Although the crashes occur at night (based on crash hour value >19), according to light condition records all the values are assigned either Daylight or Nan values.

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.

Ignore Nan values
Violation 1: ignore the values with 99.0
Violation 2: use default as Nan if the values are 0.0
Violation 3: If the crash hour is >19 interpolate the light condition's value to darkness or if crash hour value is between 12 to 19 assign Dawn or Dusk accordingly.
If a record has all values as Nan/ empty discard the record from the table.

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?

There is a lot of inaccurate data in the table. Few columns are dependent on the other columns - the number of Nan records in these columns is equal to a number of Nan records in the other table.

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