DataEng: Data Validation Activity

Priyanka Pandey

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".
    a. Crash_ID should not be empty.
    b. Vehicle_ID should not be empty.
    c. Crash Month should be 2019 for all of the records, since that is the way to know whether the data is from year 2019 or not.

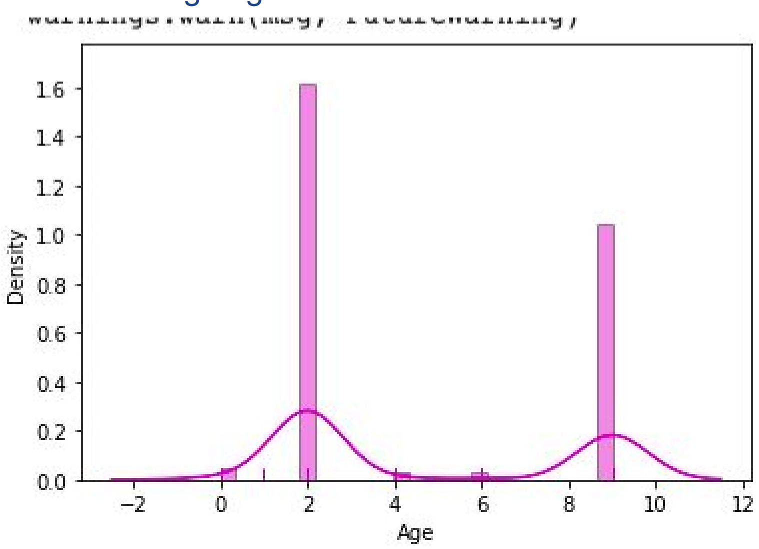
2. Create 2+ *limit* assertions. The values of most numeric fields should fall within a valid range. Example: “the date field should be between 1/1/2019 and 12/31/2019 inclusive”
    a. Crash month must be a valid month number (01-12)
    b. County code must be between (01-36).
    c. Week Day code must be between (1-7)
3. Create 2+ *intra-record check* assertions.
    a. Combination of month, day and year should be a valid date.
    b. Combination of Serial Number / County / Year must be unique.
4. Create 2+ *inter-record check* assertions.
    a. Crash ID should not be exceeding 8 characters.
    b. School Zone is a one-digit code that indicates the crash occurred should be blank, 0,1 or 9
5. Create 2+ *summary* assertions. Example: “every crash has a unique ID”
    a. HighWay number should be 26.
    b. Participant Age must be eligible for driving
    c. Age must be two-digit numeric between 00 and 99 inclusive

6. Create 2+ referential integrity insertions. Example “every crash participant has a Crash ID of a known crash”
    a. Every crash participant has a valid vehicle ID.
    b. Every crash participant has a valid participant ID.
7. Create 2+ *statistical distribution assertions*. Example: “crashes are evenly/uniformly distributed throughout the year.”
    a. Crashes are higher in number in the march month.
    b. Average age is 30.

B. Validate the Assertions

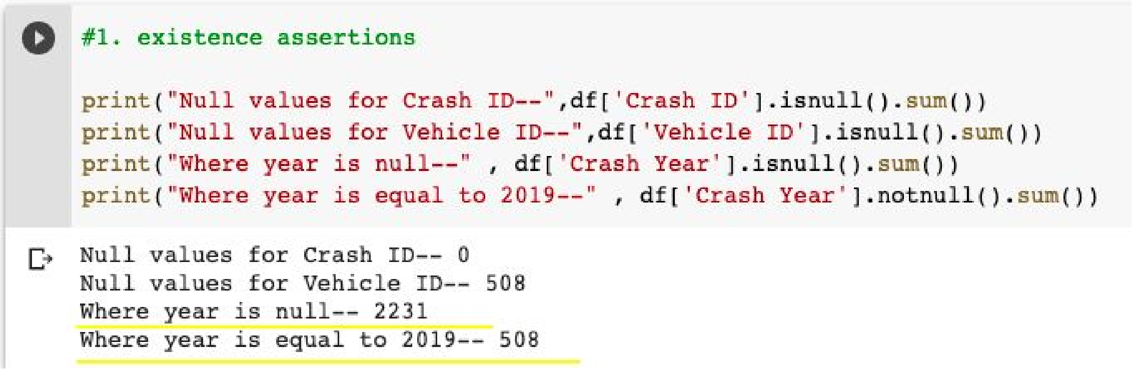
1. Now study the data in an editor or browser. If you are anything like me you will be surprised with what you find. The Oregon DOT made a mess with their data!
2. Write python code to read in the test data and parse it into python data structures. You can write your code any way you like, but we suggest 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.

a) Violation - Average age is 30.



```
warnings.warn(msg, FutureWarning)
```

b) Crash Month is null for 2231 records.



```
#1. existence assertions

print("Null values for Crash ID--",df['Crash ID'].isnull().sum())
print("Null values for Vehicle ID--",df['Vehicle ID'].isnull().sum())
print("Where year is null--" , df['Crash Year'].isnull().sum())
print("Where year is equal to 2019--" , df['Crash Year'].notnull().sum())
```

```
Null values for Crash ID-- 0
Null values for Vehicle ID-- 508
Where year is null-- 2231
Where year is equal to 2019-- 508
```

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.

a) Correctness - Age information doesn’t make sense here, so ignored. It has values like (1,2,4,6,9)
b) Correctness - added missing values for column ‘Crash Month’

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