DataEng S24: Data Validation Activity

High quality data is crucial for any data project. This week you'll gain experience with validating a real data set provided by the Oregon Department of Transportation.

Due: this Friday at 10pm PT
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.

A. [MUST] Initial Discussion Question

Discuss the following question among your working group members at the beginning of the week and place your own response(s) 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:

For a deep learning project aimed at developing a fire alarm system, I meticulously analyzed over 1200 images, focusing on how different color percentages indicated the presence of fire. I encountered challenges with varying lighting conditions that affected color accuracy. To overcome this, I refined our image preprocessing techniques, ensuring our model could reliably detect fire under diverse conditions, significantly boosting its real-world applicability.

Background

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 multi-step process.

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

B. [MUST] 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”

   Every crash record occurred in a particular location.

   Each crash record must have a highway number.

2. *limit* assertions. Example: “Every crash occurred during year 2019”

   Every crash occurred during year 2019

3. *intra-record* assertions. Example: “If a crash record has a latitude coordinate then it should also have a longitude coordinate”

   If a Crash record has a valid Crash type, then it should have valid crash severity and collision type.

4. Create 2+ *inter-record check* assertions. Example: “Every vehicle listed in the crash data was part of a known crash”

   If Driver Age is between 01 and 13, Driver License Status must = “0”

   When the Participant's Injury Severity is 7, the Participant Age must be 01 - 04

5. Create 2+ *summary* assertions. Example: “There were thousands of crashes but not millions”

Total number of crashes fall within thousands of range.

Crash Severity indicates at least one Participant was injured, but no Participant was coded

6. Create 2+ *statistical distribution assertions*. Example: “crashes are evenly/uniformly distributed throughout the months of the year.”

   Crashes have a correlation between participant age and injury severity.

   explore the distribution of crashes across severity levels.

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

C. [MUST] 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.

D. [MUST] Run Your Code and Analyze the Results

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

- In Intra record assertions, some did not meet the conditions for Crash Severity, Collision Type, and Crash Type.
  Validations Done:
  if Crash Severity is "2", "4", or "5", if Collision Type is in the range "0" to "9", "-", and "&", if Crash Type is in the range "A" to "J", "0" to "4", "6" to "9", and "&"
- In Inter record assertions, Some records did not meet the conditions for driver age and driver license status.
  Validations Done:
  if Driver Age is between 1 and 13, if Driver License Status is "0" and if conditions are met for every record
- In Inter record assertions, Some records did not meet the condition for Age format.
  Validations Done:

if Age is a two-digit numeric between 00 and 99 inclusive and if conditions are met for every record

- In Summary assertions, Some records indicate a fatal crash, but no participant was coded with a fatal injury.
  Validations Done:
  Filter records where Crash Severity indicates a fatal crash, if there are any fatal crashes where no participant was coded with a fatal injury

All the codebases are executed in jupyter notebook and pushed to github repo.

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

To resolve the violation.

Check if Crash Severity is "2","4","5" and Collision Type is "0" to "9" , "-" and "&" and Crash Type is A to J and 0 to 9 and "&". And provide valid values to make meaning value out of the same.

Conditions for driver age and driver license status are not met. So we can discard the violating rows.

Records did not meet the condition for Age format. So one can revise the age to properly adher to conditions

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

E. [SHOULD] 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.

F. [ASPIRE] 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?

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