DataEng S23: 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.

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:Yes, I have. Our team noticed that, for a particular record, values did not match between two different tables where they were supposed to. Then we looked into it further and found out that it was not the only record, there were more. We informed the team which handles data of those tables and they took care of it.

Response 2:During my under graduate I worked on data science project prediction of cancer data set is divided into 30% for testing and 70% for training. We found some errors in data includes missing values, incorrect labels. For missing values we used mean, median methods and we removed the incorrect labels which makes data set good for analysis.

Response 3:I have worked with a dataset from kaggle which has a variety of values, data types and extra characters which are not related to the database and it was also a mix of 2-3 database tables and I started cleaning it by removing the unwanted data and noise.

Response 4:While coordinating with other teams , we really didn't know the significance of each data value they asked us to populate for them. We ended up populating null instead of 0, which caused a data mapping error on their end. So we had to run a daily script to populate 0 instead of null based on a few conditions till we sent the fix to production.

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"
   For every crash there should be at least one vehicle involved

2. *limit* assertions. Example: "Every crash occurred during year 2019"
   Every crash should have crash hour between 1 to 24
   Latitude degrees should be between 41 to 47 inclusive
3. *intra-record* assertions. Example: "If a crash record has a latitude coordinate then it should also have a longitude coordinate"
   Latitude Minutes must be "null" when Latitude Degrees is "null"
4. Create 2+ *inter-record check* assertions. Example: "Every vehicle listed in the crash data was part of a known crash"

   The "Crash ID" field for each record should be unique and non-empty
5. Create 2+ *summary* assertions. Example: "There were thousands of crashes but not millions"
   Few accidents were caused by underage drivers

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

   Distribution of crashes between school zone and work zone

   The average time to respond to a crash should be within a certain range.

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 was trying to work on age criteria to validate valid age of driver, but what I saw for all the participants age is single digit which is not right way to represent the age. As per documentation 02-98 stands for anywhere between 2 years to 98 years. So we cannot conclude anything from that column.
- So many null Values are there which needs to be removed.
- I also observed that there are multiple records for one distinct pair of crash and vehicle id , which has alternative columns populated. Ideally it can be combined into one single row which will have all columns populated.

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

- add missing values- populated data with default criteria. Such as for hour column if blank populate it with 99, which suggests hour is now known at which crash has happened.
- Interpolate-predicted some of the missing and inconsistent data and interpolated it to support the claims.
- use defaults- same as add missing values

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?

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.