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: Usually data sources are different while working on a project, we have encountered when data is recorded incorrectly or entered into the system incorrectly, leading to incorrect data. Data entry errors can be fixed by reviewing the data entry process to identify the source of the error and correcting the entry.

Response 2: In a machine learning project, I was working with a dataset that had some missing values. To detect these errors, I used a missing data analysis technique to identify the missing values in the dataset. I then applied data imputation techniques to replace the missing values with either mean, median or mode values depending on the data type.

Response 3: I discovered that some data points in my dataset were outliers. To identify these errors, I used visualization. To fix these outliers, I removed or replaced them with the mean or median value.

Response 4: When we have several data sources from different regions of the globe it is very likely to encounter formatting errors. Example- Date formatting. We can fix this by changing it to one constant format.

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:
    a. "Every crash occurred on a date".
    b. Every crash record should have a Crash ID

2. *limit* assertions. Example:
    a. "Every crash occurred during year 2019."
    b. Latitude degrees and Longitude degrees coordinates for all crashes should fall within a certain range (-90,90) & (-180,180)

3. *intra-record* assertions. Example:
    a. "If a crash record has a latitude coordinate, then it should also have a longitude coordinate."
    b. *If a crash record has a Participant ID, it should also have a Participant Display Seq*

4. Create 2+ *inter-record check* assertions. Example:
    a. Every vehicle listed in the crash data was part of a known crash.
    b. Every Participant ID in the dataset should be associated with a valid Vehicle ID.
    c. Every Vehicle ID in the dataset should be associated with a valid Crash ID

5. Create *summary* assertions. Example:
    a. There were thousands of crashes but not millions"

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

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!.
   Yes, Done.

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 panda Dataframe.
   pd.read_csv("data_file.csv")

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.
   Created an ipynb file and uploaded it in git.

4. If needed, update your assertions, or create new assertions based on your analysis of the data.
   Yes.

C. Run Your Code and Analyze the Results

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

1. *existence* assertions. Example:
    a. Every crash occurred on a date.
       Yes, we have null values. We can't ignore them because the count is too much.
    b. Every crash record should have a Crash ID
       True

2. *limit* assertions. Example:
    a. "Every crash occurred during year 2019."
       The data we have is only for 2019 but because of Null values we face this. We can ignore it.

   b. Latitude degrees and Longitude degrees coordinates for all crashes should fall within a certain range (-90,90) & (-180,180)
      True

3. *intra-record* assertions. Example:
   a. "If a crash record has a latitude coordinate, then it should also have a longitude coordinate."
      True
   b. If a crash record has a Participant ID, it should also have a Participant Display Seq
      True

4. Create 2+ *inter-record check* assertions. Example:
   a. Every vehicle listed in the crash data was part of a known crash.
      Not every vehicle listed in the crash data was part of a known crash.
   b. Every Participant ID in the dataset should be associated with a valid Vehicle ID.
      Vehicle ID nan is associated with 0 Crash IDs. This data is messed up mainly because of null values.
   c. Every Vehicle ID in the dataset should be associated with a valid Crash ID
      Participant ID nan is associated with 0 Vehicle IDs

5. Create *summary* assertions. Example:
   a. There were thousands of crashes but not millions"
      True

6. Create *statistical distribution assertions*. Example:
   a. "crashes are evenly/uniformly distributed throughout the months of the year."
      Crashes are not evenly distributed throughout the months of the year.

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.

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.

Learnings -
The data is difficult to understand but at least understood few points :
We have record type column which has values as 1,2,or 3
If you filter the data for Record type == 1 The filtered data is about crash information like crash day, crash month, crash year, longitude, latitude etc.
If you filter the data for Record type == 2 The filtered data is about the vehicle information vehicle id and vehicle coded sequence.
If you filter the data for Record type == 3 The filtered data is about participants as in people involved in the crash.

Every record will at least have record type 1 and 2 but 3 is optional.

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.