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**:*
Prathamesh Chakote:
I worked on the House price forecast dataset, and there were many nil values in columns like the number of rooms, latitude and longitude, and so on. So, to deal with null values, I eliminated some rows that I believe are unimportant, and I also used Python modules like MICE. While developing a machine learning model, I received less accurate predictions; however, after cleaning and validating the data, the accuracy of my machine learning model improved.

Response:
Dileep Kumar Boyapati:
I have worked on the Big bash league dataset. In that dataset I found many null values and missing values in the columns like dismissal_kind, player_dismissed and extras_type etc. To handle these null values and missing values, I deleted the unwanted rows that are not required. After cleaning the data, the model performance got improved.

Response:

Sai Krishna Mukund

One common issue was dealing with missing values in the dataset. Missing data can affect the performance of machine learning models, so I had to decide on appropriate strategies for handling missing values, such as imputation techniques or removing incomplete records.

As mentioned earlier, the dataset had an imbalance between stroke and non-stroke instances, with a higher proportion of non-stroke cases. Addressing this class imbalance required implementing techniques like SMOTE (Synthetic Minority Over-sampling Technique) to oversample the minority class and ensure balanced training data.

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”
   “**Each vehicle involved in a crash is linked to a distinct Crash ID”.**
2. limit assertions. Example: “Every crash occurred during year 2019”
   ”Verifying if the crash ID comprises eight characters”.
   “The speed limit posted must be within the customary legal speed range enforced in the area”.

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

   "If the weather conditions indicate clear skies, then the road surface should not exhibit icy or snowy conditions."

4. Create 2+ *inter-record check* assertions. Example: “Every vehicle listed in the crash data was part of a known crash”
   "Each individual involved in the collision has had their driver's license revoked."
   "Vehicles engaged in a collision but without impact on an object during the event."
   "Every Crash ID will have a minimum of two distinct types of records linked to it."
5. Create 2+ *summary* assertions. Example: “There were thousands of crashes but not millions”
   "The majority of incidents occur at intersections and along straight roadways, indicating that these standard road configurations are pivotal areas for traffic safety."
   "Each CrashID is distinct and does not repeat across all records."
6. Create 2+ *statistical distribution assertions*. Example: “crashes are evenly/uniformly distributed throughout the months of the year.”
   "In the majority of accidents, the majority of injuries tend to be classified as minor."
   "Crashes are more frequent during midday."
   "Less than 5% of crash participants possess a suspended, revoked, or otherwise invalid driver's license, indicating that the majority of drivers involved in accidents hold valid licenses."

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:

- **"Statistically, a significant portion of individuals involved in crashes have had their driver's licenses suspended."**

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.

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.