Capstone Project 1 – Data Wrangling

The dataset chosen to use for my capstone project is the complaints flat file found here (<https://www-odi.nhtsa.dot.gov/downloads/>). Please see Appendix A under “Import Instructions” for explanations of data fields used in this document.

A new flat file is uploaded daily to the NHTSA website and that file is downloaded via ETL process daily and saved in a PostgreSQL database. Thus, a table is created for each day’s file and retained. These files are used as a version check because with new complaints that come in on a daily basis are appended to the preceding day’s file and re-uploaded by NHTSA. For this project, only BMW complaints are being used initially, but the analysis will be expanded to other manufacturers as well once the initial model is created and tested.

A few methods of cleaning are necessary in order to format the data properly. The data does not have full VINs of the vehicle complained about by a specific customer. The VIN column in the data is supposed to be the first 11 digits of the full 17 digit VIN. However, while the majority of the VINs retained are the first 11 digits, a portion of the VINs are nulls, the last 11 digits of the VIN, or another assortment of characters. These are all retained and what is usable is employed in a later step for mapping extra fields of data used in the analysis.

Additionally, with text fields such as mfr\_name, maketxt, and modeltxt, the fields are not often standardized. Because the fields on the web form that receive this data used to be text fields, not drop down menus (like currently) that limit the selection to specific entries, a person in previous years could enter a VIN that maps to a specific make and model, or enter it in manually. This created many inconsistencies. For example, a person can choose to enter their car as a BMW 3-series, or be more specific by saying it is a BMW 335. They may also leave out other details; it could be a 335i or a 335xi (all-wheel drive) and they may choose to enter this option as either 335ix, 335xi, 335 x-drive or any other potential name. Using all these combinations to create subgroups of models becomes very messy. It was also possible for the customer to enter the VIN in incorrectly by skipping a character or replacing it with something else. Because of the possible inconsistencies, VIN decoding is used instead of the multiple fields in the flat file. The first 11 digits of the VIN provide enough information as to the make and chassis code of the vehicle. The model year is also gained from other fields. Using a script with multiple conditions for decoding the VIN gives us our cleaned model name and although it is not perfect (100% of the VINs are not usable), it is much easier than trying to standardize every model name. Model names can be used as an additional step, after the VIN decoding. Manufacturer names have a similar inconsistency. Sometimes, in the data, BMW is used, or its German name Bayerische Motoren Werke. Both have to be used to narrow down the group of complaints. A similar problem was noted with Model Year: if the customer does not know the model year of their vehicle, they have the option of entering it as “9999”. For this, we must also use VIN decoding (http://www.autocheck.com/vehiclehistory/autocheck/en/vinbasics), but when an incorrect VIN is provided, we have no option to apply these fields to the data. Because our main concern is the narrative text of the complaint (cdescr), with additional features such as the make, model, model year, fires, crashes, injuries, deaths, complaint type and component description, these fields were prioritized in cleaning.

It is also possible for a person to log a complaint with the same narrative text, but with multiple component descriptions. For example, if a person wants to complain about their air-bag seat detection mat because it is not working, they may log the complaint under the following fields: AIRBAGS, SEATS, SEAT BELTS, and ELECTRONICS. This creates four complaints in the data set that all have the same odino (ODI number), however, they are four different rows. Thus, the distinction has to be made between unique odino versus unque odino AND compdesc (component description). Also, this makes it necessary to apply our own index for each row and not rely on the CMPLID (NHTSA’s unique ID which as they say, could “potentially change from one output file to the next”). When training the clustering model this will be imperative to differentiate between. Additionally, it was observed that because the customer selects these fields, they are not always correct. A customer may select fields like in the example above because they are somewhat related to the problem even though from an engineering perspective, the occupant detection mat is only in the SEAT and thus only relevant to that component description. A customer may also select multiple fields because they think that by doing so, it will escalate the severity of their complaint and get more attention on their issue from NHTSA. This creates some text descriptions that are falsely matched to a component description. This may introduce some error if they are used as a feature to train on in the clustering model. If not, and only the narrative text is used, it may prevent possible mismatches.