Radius data science assignment
Dave Kielpinski

In this assignment, I checked the validity of information in a dataset of 1 million rows using Python 3.6.3 and pandas 0.20.3.

A guiding principle of my analysis is that I should carefully avoid claiming data is valid when it isn't (i,e., false positives). Customers are likely to be annoyed by a larger dataset with false information, and would prefer a smaller dataset with less information.

Non-null values in the data are easily counted using the pandas.count method. There are relatively few null values, except for the *phone* field, for which nearly half the entries are null. The cardinality is similarly easily measured using the pandas.unique method. The instructions don't say whether to measure cardinality on the raw data or on the validated data, so the cardinality is given for the original raw data.

The data contains many duplicate records, since many rows have duplicate entries in the *name* field. However, no rows are duplicated in full. Therefore, rows that have the same *name* entry must be assumed to be entirely invalid in order to avoid false positives. After this process, we're left with 844,268 valid rows.

There were quite a few odd or unusual features in the data, which are described in the notes in Table 1 and in the discussion below. Phone numbers seem to be especially difficult to acquire.

Table 1 shows the overview of non-null values, true values, and cardinality for the various data fields, and includes notes on the validation technique. Non-null values are counted in the raw dataset (1,000,000 rows), while true values are counted only on the deduplicated data (844,268 rows).

Table 1. Overview of data fields.

field	non-null values	true values	cardinality	notes on validation
name	999986	844268	890724	All rows with duplicate names are removed in their entirety
address	999986	844194	892121	Primitive address parsing, remove anything that doesn't begin with a number
city	999986	771446	13721	Invalid values simply seem to be missing from GeoNames. Why?
state	999986	771446	60	All true values are two-letter abbreviations - no full names.
zip	999988	771446	26398	Almost all invalid values have the wrong number of digits.
time_in_business	916125	773305	12	
phone	590889	491040	575155	Many variant formats - true values can still be extracted by regex.
category_code	999986	844196	1185	Remarkably high validity rate.
headcount	962352	812404	16	Filtered out '0', 'none', 'null'
revenue	943092	796113	18	Filtered out '0', 'none', 'null'

I'll now explain the approach for validating each field in more detail.

In the *name* field, names are taken to be strings, unless they have values 'none', 'null', or ' '. It is difficult to rule out strings as names. A more comprehensive exclusion list could be helpful, but is hard to develop. However, only a few invalid values showed up, so this is probably a low priority.

The *address* field is considered valid if the entry begins with a number (like all US addresses, but unlike some foreign addresses). This validation procedure could be much improved by using a third-party parser package such as usaddress or comparing to addresses scraped from Google, but this would have added too much time to the assignment,

The *city, state,* and *zip* fields are validated using the GeoNames datasets (http://www.geonames.org/) Puerto Rico and the Virgin Islands are the only US territories included in the dataset, as found by examining the *state* field. These only account for a few records, but they're worth including. The validity test is an inner join between the Radius dataset and the GeoNames data. Therefore, for each row of the Radius dataset, all three fields are considered invalid unless all three fields appear in a single row of the GeoNames dataset. It is not enough, for instance, to have a *city* that appears in the GeoNames dataset if there is no *state* with that *city* in GeoNames. This validation technique removes over 70,000 invalid entries in each field. In contrast, independent validation of each column leaves about a third of these invalid entries in place.

All valid values for the *time_in_business* field contain the word ‘year’, as found by examining the unique values in the field, so validity of a *time_in_business* value is simply determined by testing whether the value contains ‘year’.

The *phone* field is validated by extracting the numerical value of the field, using a regex that allows for variant formatting of phone numbers. The field value is considered valid if the numerical value is a ten-digit number. This could be improved by, e.g., matching area codes to an external database.

In the *category_code* field, we find 8-digit integers, which are not part of the NAICS system proper. The extra two digits give further industry-specific information and are therefore ignored for the NAICS comparison (see https://www.census.gov/eos/www/naics/faqs/faqs.html#q6). Many of the 6-digit integer stems end with pairs of trailing zeros, and these appear to refer to shorter NAICS codes of 2 to 5 digits. A category code is deemed to be valid if a match can be found with a shorter NAICS code by iteratively stripping trailing zeros. Note that the NAICS data refers to multiple two-digit codes as ’48-49’ in a single row. Such two-digit codes are separately identified for matching.

The *headcount* field is treated similarly to the *revenue* field, with exclusion of ambiguous data.

In the *revenue* field, the validity of ‘0’, ‘none’, and ‘null’ are ambiguous - these could all indicate that the company has zero revenue, or they could simply be junk data. I chose to mark these values as invalid for the following reasons. I find the ’Less than $500,000’ value over 300,000 times in the *revenue* field, while these other values occur only 49 times. It seems as though zero-revenue businesses are normally given the ’Less than $500,000’ value by the data source.