Focaldata Technical Test

Junior Applied Data Scientist - September 2020

Scenario

One of our clients has asked us to build a model to predict vote intention and has provided us with the raw data. Before we can fire up the pipeline the raw survey must be pre-processed; the data must be restructured to the required format and we must also throw away any respondents we think are of poor quality.

The task

1. Provide us with a script (or scripts) which takes as its input the raw survey data and generates as its output a "clean" version, according to the specification in the appendix.
2. Append a new column to the "cleaned" survey, which is a flag to indicate which respondents you suspect may be of poor quality. Give this column a meaningful name (e.g. "`is_bad_respondent`"). More details on this also in the appendix.

You have been provided with

- `raw_survey.csv` - wide-format table containing raw survey responses
- `postcode_lookup.csv` - lookup table to convert postcode => constituency code.

What you should provide us with

A zipped folder containing the following:

1. A script (or scripts) that you have written that processes the raw data to the specified format;
2. A script (or scripts) that you have written to attach the "`is_bad_respondent`" to the cleaned file;
3. Any additional files / documents needed to re-execute your analysis. At the very least this should be a README.md, better still a makefile.

We mainly use Python / R for data science here (which the role requires you are comfortable with), but if you did wish to use another language (e.g. R) this would also be OK. Furthermore, the two tasks could be part of a single script if you wished - it isn't critical they are separate.

The exercise has been designed to be relatively straightforward technically, meaning that you can focus more on carrying it out to as high a quality as possible. Try and focus on the following -

- **Code quality:** are there best practises in the language you have chosen - does your code conform to it? Is it easy to understand what your code is doing?
- **Reproducibility:** will we be able to re-execute your script easily when we receive it? Have the dependencies (a `requirements.txt`, say specifying the exact package versions) been set out clearly, and any steps necessary to set these up?

- Validation: how can you be certain that your code has done the correct thing? Is it possible to include final checks / circuit breakers which trip up if something weird has been inadvertently introduced into the data?

Try and spend 2-3 hours on this exercise. This is around the amount of time we would expect you to spend on such a task as a new junior data scientist. As you get more experience this should take significantly less time. We're more looking for your approach and thinking than perfect work, so don't spend any longer than this - a well-thought out 80% complete test would be acceptable.

Please email your zipped folder back to Tamsin. Technical tests are assessed as they are returned and we're hiring on a rolling basis - best to send back as quickly as you can so you're assessed earlier, ideally in a few days.

Finally, if you have any issues with the test (e.g. it's taking a lot longer than expected, especially for one small task), feel free to check in with us and see if there's a way around it. We encourage candidate feedback - if it takes you 5 hours, others might be the same, which isn't what we want to happen!

Appendix

Cleaning specification notes

Your cleaning script should do (at least) the following.

1. The data science pipeline expects that NULL values are coded in a consistent fashion. Transform all missing values such that they are consistent with one another.
2. De-duplicate the data such that every row is unique.
3. Convert the year of birth column into an "`age`" column (i.e. `year_of_birth`: 1990 => `age`: 30)
4. Using the lookup table (`postcode_lookup.csv`) convert the postcode column into a Westminster Constituency Code.
5. You will see that certain questions are follow-ups, and only get asked according to the response to the previous question ("*If you answered yes, to the previous question - would you ever ...?*"). In these cases please combine the two such that the "no" response (in this case) appears in the follow-up question, rather than the follow-up column containing missing values for those people to whom the question was not presented.
6. Please remove any columns that you think look like junk.

Feel free to add any additional processing you feel is required but please make sure the justification for this is provided (either implicitly in the code / explicitly through comments or documentation).

Data quality notes

In adding a "`is_bad_respondent`" flag to a row of the survey we are stating that this respondent's responses cannot be relied upon as credible. This is fundamentally subjective, and as such you have free licence to tag the data in any way you see fit, provided your approach is reasonable and well-executed. However, as a starting point, here are some suggestions for potential indicators of low quality responses:

1. Suspiciously fast response times.
2. Inconsistency of responses - perhaps saying that you simultaneously do and do not like something, or that you live in different parts of the country on different parts of the survey.
3. Stupid responses - odd date of birth / postcode / saying you took part in a vote you would certainly have been too young to vote in / silly freetext responses, etc.

We are more interested in how you tackle the problem than the actual number of people that you do or do not flag. There is no correct answer we are checking your responses against. What we would like to see is that the way you have approached the problem is sensible and that your solution has been scripted correctly and clearly. We'd be interested in seeing your thinking and what you notice, so we recommend putting comments in the script for this and/or in any supplementary documentation (e.g. a README.md)