Alan Verdugo
2020

# Asserting data quality with Great Expectations

# What?

What is “Great Expectations”?

- A book by Charles Dickens.
- FOSS Python module to help with data cleansing and processing.
- A relatively young project (since February, 2018. Current version: 0.11.7).
- Try it now: pip install great_expectations --user

What is an “Expectation”?

- Expectations provide a flexible, declarative language for describing expected behavior.
- Unlike traditional unit tests, Great Expectations applies Expectations to data instead of code.

Source If Applicable

# What?

- Great Expectations helps teams save time and promote analytic integrity by offering a unique approach to automated testing: pipeline tests.
- Pipeline tests are applied to data (instead of code) and at batch time (instead of compile or deploy time).
- Pipeline tests are like unit tests for datasets: they help you guard against upstream data changes and monitor data quality.
- Software developers have long known that automated testing is essential for managing complex codebases.
- Great Expectations brings the same discipline, confidence, and acceleration to data science and engineering teams.

Source If Applicable

# Why?

- Data pipelines often become deep stacks of unverified assumptions.
- Mysterious (and sometimes embarrassing) bugs crop up more and more frequently.
- Resolving them requires painstaking exploration of upstream data, often leading to frustrating negotiations about data specs across teams.

- Examples:
  - The data format has changed.
  - The data is the same, but the extraction has been modified.
  - We incorrectly *assume* we should be *expecting* some specific format.

Source If Applicable

# Why?

- Save time during data cleaning and munging.
- Accelerate ETL and data normalization.
- Streamline knowledge capture and requirements gathering from subject-matter experts.
- **Monitor data quality in production data pipelines and data products.**
- Automate verification of new data deliveries from vendors and other teams.
- **Simplify debugging data pipelines if (when) they break.**
- Codify assumptions used to build models when sharing with other teams or analysts.
- Develop rich, shared data documentation in the course of normal work.
- **Make implicit knowledge explicit.**
- … and much more

Source If Applicable

# Sounds familiar?

- Empty result sets that should not be empty.
- Data formatting issues. For example:
  - DD/MM/YYYY instead of DD-MM-YYYY
  - DD-MM-YYYY instead of DD-MM-YYYY 00:00:00
  - “NULL” instead of “ ”, or viceversa.
  - “1” (str) instead of 1 (int) or 1.0 (float).
- Unexpected, but similar data. For example:
  - You were expecting **“US”** or **“MX”** and you get **“USA”** or **“Mexico”.**
  - You were expecting **“IBM”** and you get **“IBM Corp.”**, or **“International Business Machines.**
  - You were expecting **“True”** or **“False”**, but got **“T”**, **“F”**, **“TRUE”**, **“FALSE”**, **“1”**, **“0”**...
  - Etc.
- In short: Somebody messed up and now you have to find the problem and fix it, *quickly*.

# Great expectations

DEMO

# Great expectations

The idea behind Great Expectations is that **each dataset should be paired with a set of expectations—they are tests for data during runtime.**

The library **does not throw an error** if a new batch of data in the dataset violates the expectations, as how I want to use this piece of information depends on the context. Depending on the situation, I might:

- Reject the new batch of data, and only use the old and validated ones.
- Raise a warning in the logs.
- Send an email/phone call at 3 AM to the person on call to take a look.

...But it is unlikely that I want to throw an error and stop the whole process.

Source If Applicable

# Great expectations

There are MANY default expectations…

https://docs.greatexpectations.io/en/0.7.11/glossary.html

Also:

- Custom expectations.
- Integration with Pandas, Spark, SQLAlchemy, Postgres, MySQL, BigQuery, S3, Airflow, etc.
- “Tests are docs and docs are tests” (Beta).
- Automated data profiling (Experimental).

Source If Applicable

# References

- Official documentation:
  https://docs.greatexpectations.io/en/latest/
- Down with Pipeline debt / Introducing Great Expectations:
  https://greatexpectations.io/blog/down-with-pipeline-debt-introducing-great-expectations/
- Up & Running with Great Expectations:
  https://ianwhitestone.work/hello-great-expectations/
- Great Expectations: Validating datasets in machine learning pipelines:
  https://blog.codecentric.de/en/2020/02/great-expectations-validating-datasets-in-machine-learning-pipeline/
- Document Analysis as Python Code with Great Expectations:
  https://changhsinlee.com/python-great-expectations/

Source If Applicable