Data and Feature Debugging

Low-quality data will significantly affect your model's performance. It's much easier to detect low-quality data at input instead of guessing at its existence after your model predicts badly. Monitor your data by following the advice in this section.

Validate Input Data Using a Data Schema

To monitor your data, you should continuously check your data against expected statistical values by writing rules that the data must satisfy. This collection of rules is called a **data schema**. Define a data schema by following these steps:

1. For your feature data, understand the range and distribution. For categorical features, understand the set of possible values.
2. Encode your understanding into rules defined in the schema. Examples of rules are:
    - Ensure that user-submitted ratings are always between 1 and 5.
    - Check that “the” occurs most frequently (for an English text feature).
    - Check that categorical features have values from a fixed set.
3. Test your data against the data schema. Your schema should catch data errors such as:
    - anomalies
    - unexpected values of categorical variables
    - unexpected data distributions

Ensure Splits are Good Quality

Your test and training splits must be equally representative of your input data. If the test and training splits are statistically different, then training data will not help predict the test data. To learn how to sample and split data, see the Sampling and Splitting Data (https://developers.google.com/machine-learning/data-prep/construct/sampling-splitting/sampling) section in the Data Preparation and Feature Engineering in ML course.

Monitor the statistical properties of your splits. If the properties diverge, raise a flag. Further, test that the ratio of examples in each split stays constant. For example, if your

data is split 80:20, that ratio should not change.

Test Engineered Data

While your raw data might be valid, your model only sees engineered feature data. Because engineered data looks very different from raw input data, you need to check engineered data separately. Based on your understanding of your engineered data, write unit tests. For example, you can write unit tests to check for the following conditions:

- All numeric features are scaled, for example, between 0 and 1.
- One-hot encoded vectors only contain a single 1 and N-1 zeroes.
- Missing data is replaced by mean or default values.
- Data distributions after transformation conform to expectations. For example, if you've normalized using z-scores, the mean of the z-scores is 0.
- Outliers are handled, such as by scaling or clipping.

Previous
← Programming Exercise (/machine-learning/testing-debugging/common/programming-exercise-debugging-challenges)

Next
Model Debugging (/machine-learning/testing-debugging/common/model-errors) →

Last updated 2019-03-06 UTC.