Is My Data Any Good?
A Pre-ML Checklist

Google Cloud

About the checklist

This document consists of a list of guidelines and rules of thumb for assessing and preparing data for a machine learning (ML) project. The intended audience is anyone involved with such a project, including technical teams and business decision makers. If you are thinking about collecting data that can later be used for ML, this document can help you understand what to collect. If you already have data, it can help you evaluate and prepare it for ML. This document contains some of the more common topics and is by no means exhaustive.

Document all your data preparation steps (assumptions, transformations, code, and so on). As much as possible, keep the original (raw) data intact. Do not replace raw data with data transformations (for example, inferred missing values, corrected wrong values). This way, you can go back and correct things if you make any mistakes.

Terms you should know

You'll need to know the following terms to understand this document:

- Supervised learning: The ML task of learning a function that maps an input to an output based on example input-output pairs. For example, learning a function to classify an email as one of two possibilities — "is spam" or "is not spam" — is a supervised learning task. Unless otherwise indicated, this checklist discusses supervised ML.
- Example: Each input-output pair. Also called a *sample*.
- Machine learning model: A function that maps an input to an output.
- Train a machine learning model: Learn the function that maps an input to an output based on examples.
- Training data: A set of input-output pairs used to train an ML model.
- Input: The thing you want to make a prediction about. Also called an *instance*. For example, in the spam classification problem, an email is an instance.
- Output: The thing that you want to predict. Also called the *target* or *response*.
- Feature: An input attribute used for prediction. For example, in the spam classification problem, one feature can be whether an email contains the recipient's name.
- Label: The value of the target. This can be either the predicted value generated by ML or the actual value reflective of the real world. For example, in the spam classification problem, if you have an email that has been verified as spam, you can use this email as an example with the label "is spam."
- Field: An item of data. To do ML, a field of data must be converted to a format that can be used by the model. For example, in the spam classification problem, `email_origin_city` is a field of data that can have values such as `NY`, `SF`, and so on. To input to a model, you can convert this field to a set of features — `email_origin_city_is_NY`, `email_origin_city_is_SF`, and so on — each having a discrete numeric value of `0` ("no") or `1` ("yes") that the model can use. An input field is also called an *input variable* and an output field is also called an *output variable*.
- Classification: A supervised ML problem involving a categorical target.
- Regression: A supervised ML problem involving a numerical target.

Content

1. Essential checks

Use the following questions to determine whether your data meets the minimum requirements for ML. The answers should all be "yes." A "no" wouldn't automatically disqualify your data, but you would most likely need to discuss it with a data scientist to explore the solutions and evaluate the limitations.

1.1. General

The following items are the basic requirements and rules of thumb that apply to most ML projects involving structured (for example, tabular) and/or unstructured (for example, text) data.

☐ 1. Does the data include information that can predict the target?

For example, to detect a fraudulent credit card transaction, you'll need to have information on the transaction amount, the transaction location, the card holder primary location, and so on. Data such as the name of the cardholder is not useful.

☐ 2. Does the granularity of training and prediction match?

For example, if you want to predict the weather temperature hourly, but you have been recording the temperature once a day or once a week, your data doesn't have enough signal. Similarly, you cannot predict a very granular characteristic based on aggregates. You can build a model to convert blurry images to sharp images, but you first need to give it some sharp images as training data to learn from.

☐ 3. Do you already have labeled data (that is, training examples)?

Unless you are planning to do unsupervised learning, you will need labeled data. If you don't have any, you can try crowdsourcing services such as Google's human labeling service.

- **If you have a categorical target, you'll need some labeled data for each of the categories.**
  For example, in a spam email classification problem, you'll need some examples both of correctly identified spam emails and of correctly identified nonspam emails. You cannot build the model if you have examples for only one of the categories but not all.

- **If you have a numerical target, you'll need some labeled data for different ranges of values.**
  For example, if you want to predict the height of a person from her photo, you'll need some photos of tall persons, of average-height persons, and of short persons, all with the correct heights assigned to them.

☐ 4. Is your data correct/accurate?

You cannot expect great performance from a model trained on data when the labels and/or features of that data are incorrect or inaccurate. If you have erroneous values in your data, for example, due to software or hardware failure, try to identify and impute the correct values as much as possible. But when you correct the values, do not delete the errors or replace them. Instead, create a new field to store the corrected values. In some problems (for example, a failure forecast), data errors can be an important predictor field.

☐ 5. Do you have enough data?

You'll need to have enough data to train a model with a reasonable level of performance. There are several rules of thumb you can follow to help ensure enough data. Note that these are not mathematical statements.

- **If you are predicting a category, the number of examples you'll need for each category is 10 × number of features.**
  For example, suppose you want to categorize 10 × 10 images of handwritten digits. Each image has 100 pixels (that is, features) and there are 10 categories (0–9). Following the rule of thumb for number of examples per category, you'll need 10,000 images. With more data, you can try more complex models (for example, deep neural networks) to get better performance. If the number of samples in a specific target category is far fewer than in other categories, the ML model's learning is severely curtailed. Also remember to count a categorical input field with *k* categories as *k* - 1 features (because to encode *k* categories, you'll need *k* − 1 dummy variables). That means the more categories your input field has, the more data you'll need.

- **If you are predicting a number, you'll need 50 × number of features.**
  This statement is assuming samples are distributed evenly across the range of target values. Sample imbalance leads to requiring more data.

- **Transfer learning may require less data.**
  For example, if you are engaged in transfer learning with the preceding handwritten digit classification example, you may get highly accurate results with Cloud AutoML Vision using only 50 training images per category, that is, 500 images in total.

- **If the acquisition of sufficient labeled data is difficult, semi-supervised learning can be of great practical value.**
  In semi-supervised learning, you use a small amount of labeled data in combination with a large amount of unlabeled data to improve learning performance. Discuss this option with a data scientist to evaluate it for your project.

☐ **6. Is the data easily accessible by the team and machines performing the ML?**

If there are strict restrictions that prevent easy access, you should address them before the ML tasks start.

☐ **7. Can you read the data fast enough?**

You don't want to wait days to generate your features and/or labels each time you make a change to your data preparation pipeline. For example, suppose you have a massive database of sensor readings that can be accessed using an API. If you want to generate a new feature that is the sum of all sensor readings, you'll want to be able to do so within a reasonable amount of time. The same would be true if, a few days later, you discover that all the readings in a range of dates were flipped in value (due to a software bug), meaning that you'll need to regenerate the reading sums.

☐ **8. Do you have documentation for each field of data?**

Can you provide a brief description for each field that includes its name and type, what it represents, how its value is measured, when the data is collected, whether the original values can get updated, and what its applicability is to the use case? Do you have records of any changes in the method of collection or in the measurement of a field? (For example, switching from miles to kilometers or replacing a broken IoT sensor.)

☐ **9. Are the missing values a small percentage of the fields of interest?**

Note that missing values can be represented in different forms even for the same field: `" "`, `None`, `NULL`, `NaN`, `0`, `-1`, `9999`, and so on. The more missing values, the less useful the data. As a check, do you have enough useful data left if you consider only data with no missing values?

1.2. Forecasting

The following items apply to forecasting problems. In a forecasting problem, you predict future outcomes based on data of the past. For example, you have past sales for a product and want to predict future sales.

☐ 10. If your data is periodic, do you have data for 3 × period?

For example, if you want to forecast sales for a seasonal item such as an ice bag, a rule of thumb is that you'll need more than three years of data. If the seasonality pattern of your data is weekly, you'll need at least three weeks. Often you'll need more than 3 × period to train a model and properly evaluate its quality prior to using it to make business decisions.

☐ 11. If you want to forecast *n* periods in advance, do you have *n* + 2 periods of data?

For example, if you want to build an ML model to forecast customer spending in the next five years, you would need at least seven years of data to train and evaluate the model properly. You'll need five years to record your first labels, one more year for the model to learn the general trend (that is, increasing, decreasing, or unchanging) for customer spending, and one more year to evaluate the model.

☐ 12. Do you know the timestamp at which each data value was obtained or updated?

To build (that is, to train) a forecasting model, you must re-create snapshots of the data in the past (as examples) and show them to the model to learn from. For this reason, you'll often need to know the time of each new data value, as well as the time of each data value update (for example, if an item price is changed or if a user changes a setting). In some problems, it may be sufficient to know the chronological order the data values are obtained to re-create data snapshots. Discuss your situation with a data scientist.

2. Additional checks

Use the following items to better understand your data and set expectations appropriately. The answers to the following questions should all be "yes." Please discuss any "no" answers with a data scientist. Most "no" answers are manageable problems, but it is important that you know beforehand about the limitations to find an appropriate solution.

☐ 13. Is your data unbiased?

For example, consider a bank trying to build an ML model to improve their existing process for determining who to grant loans to. Their labeled data consists of years of information on the loan applications already granted and whether these loans defaulted. Since the bank is looking to build a model to determine from the *complete* pool of applicants which loans will default, but their data consists of only loan applications that were approved, the data is biased. That is, the bank has declined all applications that were considered too risky and there is no default information for this group for the model to learn from. Had the bank approved all the applications, the data would have been unbiased.

- **If you are considering building a model on a subset of your data, make sure that subset does not introduce bias.**
  For example, if you want to predict hotel bookings in regions across the world, but have data only for Europe, your model may not work well for other regions.

- **If you use a subset of data for training, make sure it is unbiased.**
  If you have to use a subset of your data (for example, due to regulations, privacy, data size, or limited resources) to train the model, make sure it has the same statistical distribution as the original data. For example, if you want to predict annual bookings for an international hotel chain, do not use data only from San Francisco. Try to sample randomly across the world. When you have too much data, there can be the temptation to randomly drop what feels like unnecessary excess, taking files 1-15, for example, and ignoring files 15-100. This is a mistake. Try to sample all the files. If you cannot sample them with the same probability, account for that by using weights in the training.

☐ 14. If there are missing values, do you know the causes?

Handling missing data is possible, but if the cause of the missing data is not well understood, ML may not be as successful.

☐ 15. If there are missing values, do they occur at random?

Missing values are common. For example, say you are missing the age and/or occupation of some of your customers. If the values are missing completely at random (MCAR) — that is, if there is nothing systematic happening that makes some data more likely to be missing than other data — then the available data is unbiased. For values not MCAR, there are two cases. If the missing values can be fully accounted for by other variables, this group is called missing at random (MAR). In this case, you should impute the missing values properly using the available data to build a model that is reasonably unbiased. But if the values are missing systematically, that is, missing *not* at random (MNAR), the model built on this data may be biased and so may underperform at deployment.

☐ 16. For each field (input or target), does the data have the same unit?

For example, a data field called `item_cost` has prices that can be in different currencies. All the values should be converted to one currency.

☐ 17. For each field (input or target), is the meaning of the data consistent?

For example, the data is inconsistent in a field `item_price` where some values record the wholesale price and some values instead record the retail price. If it is not possible to determine the meaning of each value, the data field may not be useful for ML. Try to make all the values consistent. If that is not possible, create a new feature that identifies the meaning (for example, wholesale price or retail price). Inconsistent sensor calibrations is another example. Studies of consistency of human responses suggest that the same person may give different answers to simple questions when asked that same question at different times.

☐ 18. Is the same value recorded in the same way everywhere?

For example, in a field `job_title`, if “Engineer” is recorded in multiple ways such as `engineer`, `Eng`, and `eng`, then the same value is not represented consistently. To improve the ML outcome, any such inconsistency should be identified and made consistent.

3. Preparing the data

The following tasks will help you prepare the data for ML. This preparation can significantly accelerate the development of the ML model. Additionally, these tasks include some of the common ways to improve the speed and performance of your ML model.

☐ 19. Integrate data from diverse input sources.

For example, paper documents, text files, image files, data warehouses, and slides can all contain useful information that you should integrate together.

☐ 20. If your data is scattered, identify and consolidate it.

For example, if the shipping address of some of your customers is stored under `mailing_address` and the rest under `shipping_address`, identify this fact and, if possible, consolidate the two into one feature. Consolidating your data can ease handling of the data and interpretability of the model. Do not consolidate, however, at the cost of losing any data. For example, if the redundant values stored in two different places do not match, do not arbitrarily drop one.

☐ 21. Identify and impute missing values.

Make sure you don't lose any information during the imputation. For example, for a feature `payment_amount` that contains both None and NA, create two new features named `payment_amount_is_None` and `payment_amount_is_NA` and record whether the feature is None or NA, respectively. This way, you capture hidden information in missing values. If None means no payment and NA means information was not recorded, the ML model learns the difference.

☐ 22. Remove all sources of noise from your data.

Treat noise similarly to errors. Identify and remove as much as possible. For example, apply notch filters to reduce periodic noise from images before doing image classification. Or, to do NLP on scanned documents, first remove any scanning artifacts and irrelevant material (for example, a copier cover page) that is not useful for the ML model.

☐ 23. Create new features that improve predicting the target.

This is called *feature engineering*. To predict which hotel a traveler will reserve on your travel website, you can create a new feature that is equal to the average spending on accommodation in her previous trips. Be careful that the new features don't lead to pitfalls such as data leakage.

☐ 24. Look for new sources of information to complement your data.

For example, if one of your features is `organization_name`, you can complement it with features containing organization information obtained from different resources, including free public datasets (for example, Google BigQuery public datasets).

☐ 25. Identify and remove all sources of data leakage.

Data leakage happens if you train your model using predictive information that won't be available at serving time (that is, prediction time). For example, in an ML competition for identifying which acoustic recordings contained calls from an endangered species of whale, the audio files with whale calls turned out to have a very specific set of file sizes. Learning that information (and some other leakage), the model did great on the competition dataset. Obviously, such a model would fail in a real test. As another example, suppose you'd like to use the electronic health records (hundreds of fields of data) to predict whether someone would need surgery. Drop the field `surgery_room_number`. Otherwise, the model can learn that whoever had a valid value for this field also had surgery.

☐ 26. Integrate all the features of an instance into one object.

For example, suppose your data is stored in a relational database across multiple tables such as user profile, account info, click history, and so on. Integrate (that is, denormalize) those tables into one single table in which each row contains one instance with all the relevant information.

☐ 27. Convert data to formats that can be read fast for training the ML model.

This can have a significant impact on the training time of your model. The TFRecord file format gives you great performance for both structured data (for example, tabular sales information) and unstructured data (for example, text, audio, image, and video).

☐ 28. For a forecasting problem, build a pipeline to easily re-create a snapshot of the data at an arbitrary time in the past.

Doing this correctly is critical for building a useful model, and in practice it is often easier said than done. For each training example, you must use the data that was available at the time you would have wanted to predict the target, if you'd had the ML model. For example, suppose you want to build a model to predict the probability of default for a loan application. For each training example (that is, defaulted or fully

repaid application), use the applicant's FICO score as entered on the loan application, not a more recent FICO score. A great way to make sure that the data seen at training time and the data seen at serving time are not skewed (that is, are not inconsistent) is to save the set of features used at serving time, and then to use those features at training time. Ideally, you'd do this for every example. But if you can't, then do it for enough examples to verify consistency between training and serving.

☐ 29. Implement data quality tests.

Create automatic tests to ensure data quality after each iteration, for example, new imputation rules, adding a new feature, redefining your labels, and so on. Some common tests include making sure that:

- ❏ There are no missing values.
- ❏ There are no erroneous values. For example, if you have a feature `ad_click_ratio` that stores the ratio of ads clicked to, make sure its value is a float between 0 and 1.
- ❏ There are no duplicates.
- ❏ There are no denormalization mistakes (for example, incorrect mappings).
- ❏ The distribution of features and the label are as expected. If you expect a specific feature to have a mean of zero, it is alarming if the mean of the values are three standard deviations away from zero.
- ❏ There is consistency between serving and training, including no data leakage.