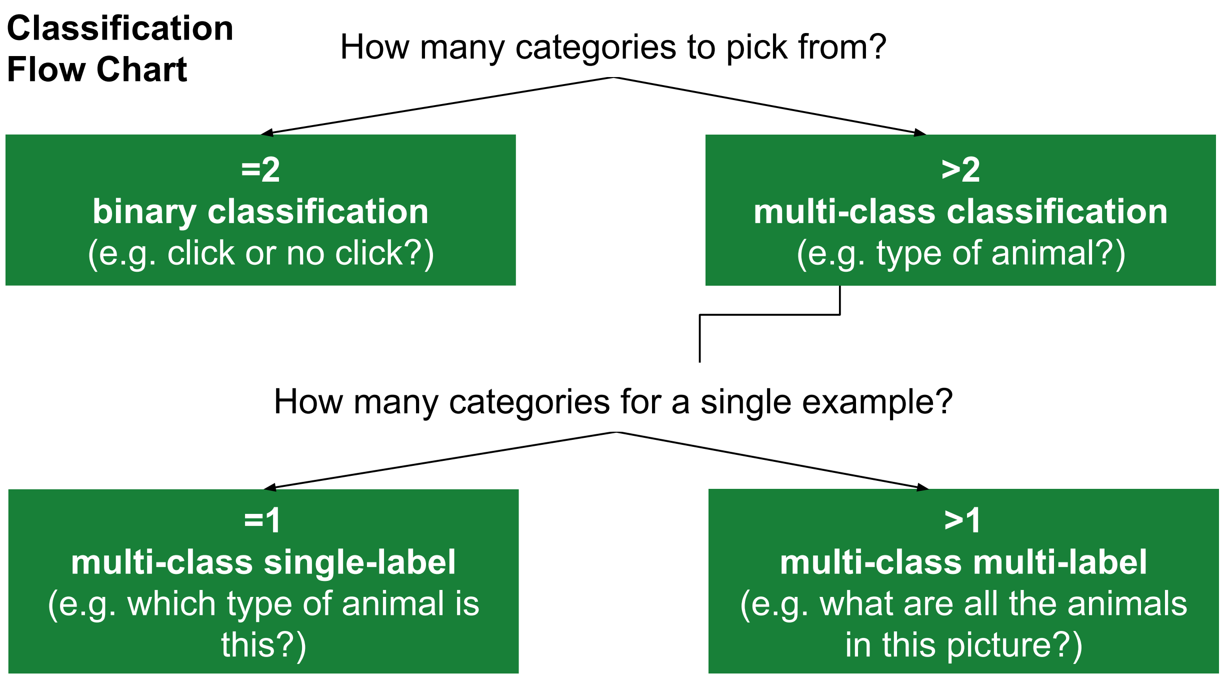
Formulate Your Problem as an ML Problem

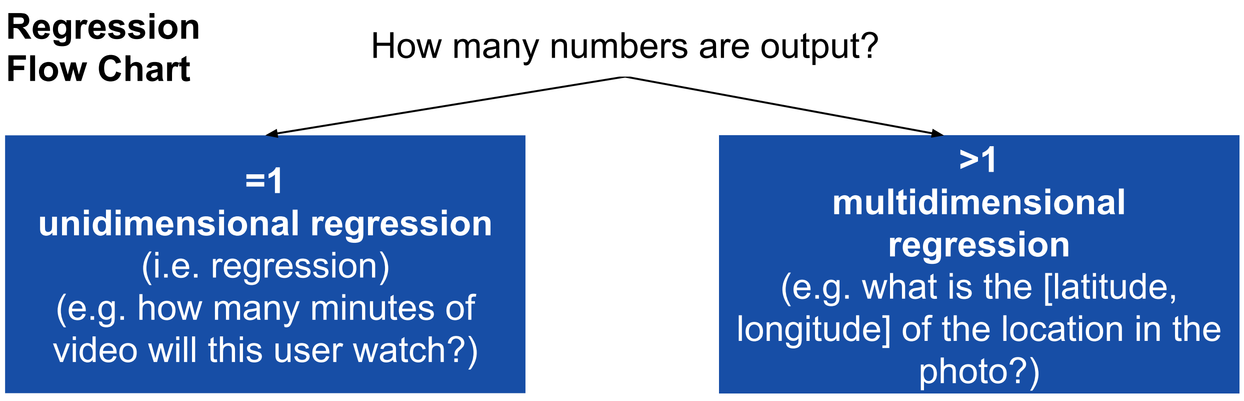
This section is a guide to the suggested approach for framing an ML problem:

1. Articulate your problem.
2. Start simple.
3. Identify Your Data Sources.
4. Design your data for the model.
5. Determine where data comes from.
6. Determine easily obtained inputs.
7. Ability to Learn.
8. Think About Potential Bias.

Articulate Your Problem

There are several subtypes of classification and regression. Use the corresponding flowchart to identify which subtype you are using. This flowchart helps you assemble the right language to discuss your problem with other ML practitioners. Use the Classification or Regression flowchart depending on your business problem.


Classification Flow Chart
How many categories to pick from?
=2 binary classification (e.g. click or no click?)
>2 multi-class classification (e.g. type of animal?)
How many categories for a single example?
=1 multi-class single-label (e.g. which type of animal is this?)
>1 multi-class multi-label (e.g. what are all the animals in this picture?)


Regression Flow Chart
How many numbers are output?
=1
unidimensional regression
(i.e. regression)
(e.g. how many minutes of video will this user watch?)
>1
multidimensional regression
(e.g. what is the [latitude, longitude] of the location in the photo?)

Our problem is best framed as:

- Binary classification
- Unidimensional regression
- Multi-class single-label classification
- Multi-class multi-label classification
- Multidimensional regression
- Clustering (unsupervised)
- Other (translation, parsing, bounding box id, etc.)

Then, after framing the problem, explain what the model will predict.

Putting each of these elements together results in a succinct problem statement, such as the following:

Example

Our problem is best framed as 3-class, single-label classification, which predicts whether a video will be in one of three classes—`{very popular, somewhat popular, not popular}`—28 days after being uploaded.

Start Simple

Can you simplify your problem?

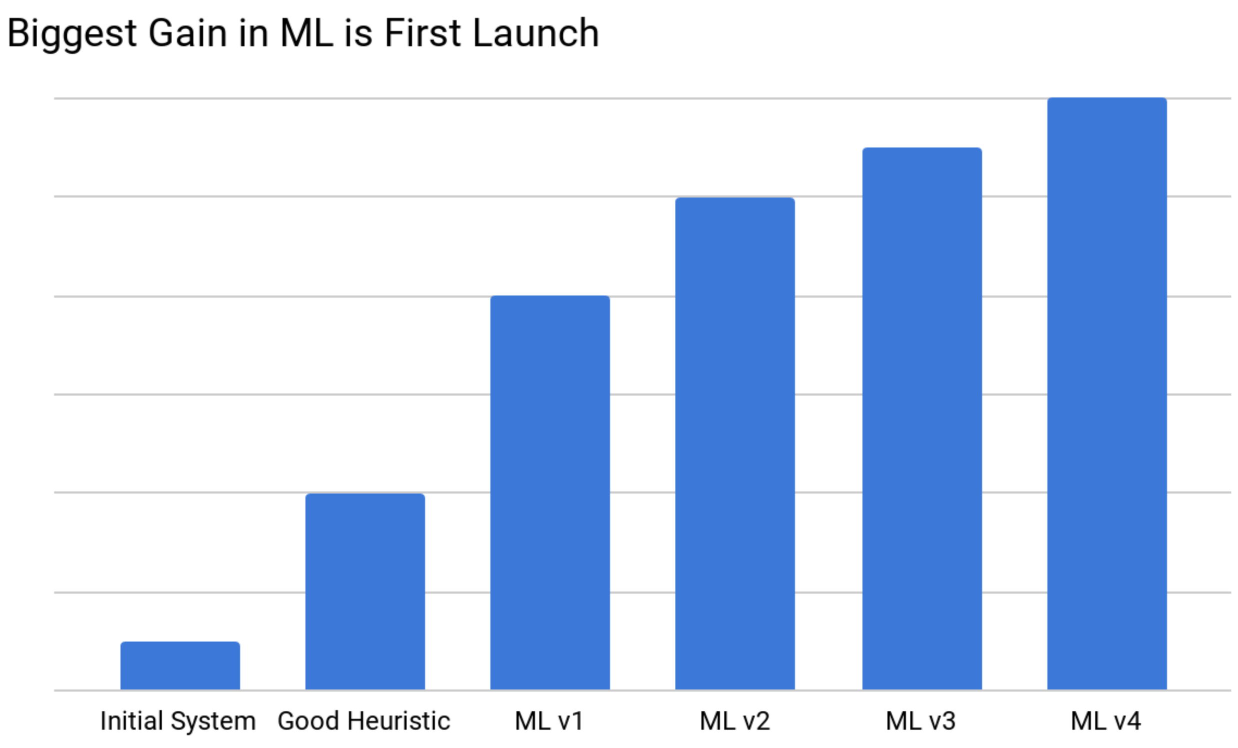
First, simplify your modeling task. State your given problem as a binary classification or a unidimensional regression problem (or both). Both problems are well-traversed, supervised approaches that have plenty of tooling and expert support to help get you started.

Then, for that task, use the simplest model possible. A simple model is easier to implement and understand. Once you have a full ML pipeline, you can iterate on the simple model with greater ease.

Examples
We will predict whether an uploaded video is likely to become popular or not (binary classification).
We will predict an uploaded video’s popularity in terms of the number of views it will receive within a 28 day window (regression).

Simple models provide a good baseline, even if you don't end up launching them. In fact, a simple model is probably better than you think. Starting simple can help you determine whether a complex model is even justified. More complex models are harder and slower to train and more difficult to understand, so stay simple unless the complexity provides a large enough improvement in model quality to justify these tradeoffs.

Ɩost of ML is on the data side. Getting a full pipeline running for a complex model is harder than iterating ɔdel itself.



The biggest gain from ML tends to be the first launch, since that's when you can first leverage your data. Further tuning still gives wins, but, generally, the biggest gain is at the start so it's good to pick well-tested methods to make the process easier.

Identify Your Data Sources

Provide answers to the following questions about your labels:

- How much labeled data do you have?
- What is the source of your label?
- Is your label closely connected to the decision you will be making?

Example
Our data set consists of 100,000 examples about past uploaded videos with popularity data and video descriptions.

Design your Data for the Model

Identify the data that your ML system should use to make predictions (input -> output), as in the following table:

Title	Channel	Upload Time	Uploader's Recent Videos	Output (label)
My silly cat	Alice	2018-03-21 08:00	Another cat video, yet another cat	Very popular
A snake video	Bob	2018-04-03 12:00	None	Not popular

Each row constitutes one piece of data for which one prediction is made. Only include information that is available at the moment the prediction is made. Each input can be a scalar or a 1-dimensional (1D) list of integers, floats, or bytes (including strings).

If an input is not a scalar or 1D list, consider whether that is the best representation for your data. For example:

- If a cell represents two or more semantically different things in a 1D list, you may wish to split these into separate inputs.
- If a cell represents a nested protocol buffer (/protocol-buffers), you may wish to flatten out each field of the nested protocol buffer.
- Exceptions: audio, image and video data, where a cell is a blob of bytes.

Tips for audio/image/video data	Examples

There may not be explicit inputs.	The only inputs may be the bytes for the audio/image/video.
There may be metadata accompanying the image.	Compression format, object bounding boxes, source
Your outputs may be simplified for an initial implementation.	Rather than doing bounding-box object detection, you may create a simple binary classifier that learns whether one type of object is present in the image or not.

Determine Where Data Comes From

Assess how much work it will be to develop a data pipeline to construct each column for a row. When does the example output become available for training purposes? If the example output is difficult to obtain, you might want to revisit your output, and examine whether you can use a different output for your model.

Make sure all your inputs are available at prediction time in exactly the format you've written down. If it will be difficult to obtain certain feature values at prediction time, omit those features from your model.

Example

We applied the labels `{very popular, somewhat popular, not popular}` to each video that fell within a determined range of views and "thumbs ups" and determined keyword descriptions for each video. Hand-generating descriptions is not sustainable, so we are considering adding a keyword description to the upload form.

Determine Easily Obtained Inputs

Pick 1-3 inputs that are easy to obtain and that you believe would produce a reasonable, initial outcome.

Which inputs would be useful for implementing heuristics mentioned previously?

Consider the engineering cost to develop a data pipeline to prepare the inputs, and the expected benefit of having each input in the model. Focus on inputs that can be obtained from a single system with a simple pipeline. Start with the minimum possible infrastructure.

Ability to Learn

Will the ML model be able to learn? List aspects of your problem that might cause difficulty learning. For example:

- The data set doesn't contain enough positive labels.
- The training data doesn't contain enough examples.
- The labels are too noisy.
- The system memorizes the training data, but has difficulty generalizing to new cases.

Example

The measure "popular" is subjective based on the audience and inconsistent across video genres. Tastes change over time, so today's "popular" video might be tomorrow's "not popular" video.

Think About Potential Bias

Many dataset are **biased** (/machine-learning/glossary#bias_ethics) in some way. These biases may adversely affect training and the predictions made. For example:

- A biased data source may not translate across multiple contexts.
- The training sets may not be representative of the ultimate users of the models and may therefore provide them with a negative experience.

Example

Since the measure "popular" is subjective, it is possible that the model will serve popular videos that reinforce unfair or biased societal views.

Previous
← Try it Yourself (/machine-learning/problem-framing/try-it/framing-exercise)

Next
Try it Yourself (/machine-learning/problem-framing/try-it/formulate-exercise) →

Last updated 2020-08-10 UTC.