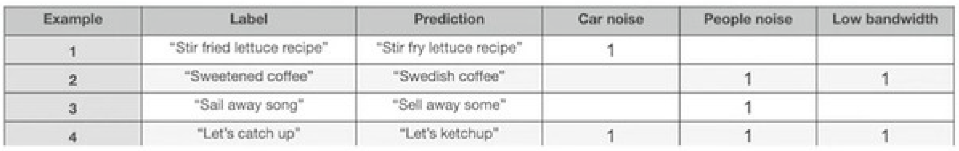
Error analysis example

The first time you train a learning algorithm, you can almost guarantee it won't work. Therefore, we can think of the heart of the ML development process as **error analysis**. It can tell you what's the most efficient use of your time in terms of what you should do to improve your learning algorithm's performance.

Example: Speech recognition: After training your model, pick a number of (say 100) mislabeled examples from the dev set. Create a spreadsheet and for each example specify your guess on why that example is mislabeled by the model (see below). ***This process helps you understand whether specific categories, that may be the source of most errors, are worthy of further effort and attention.***

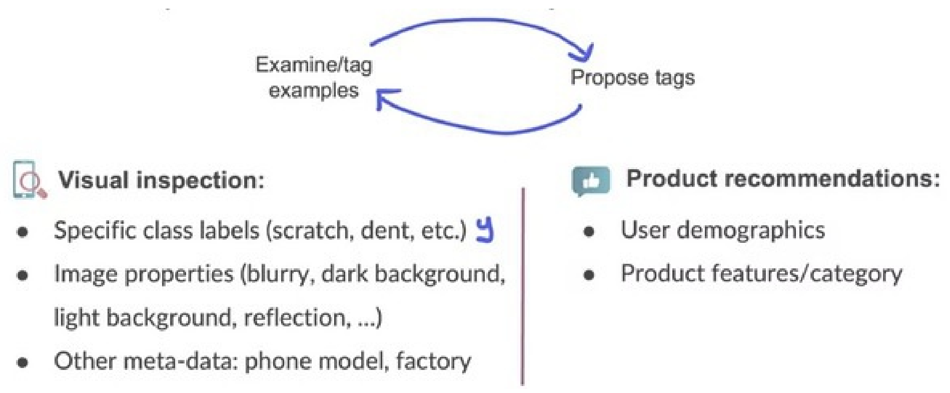


Example	Label	Prediction	Car noise	People noise	Low bandwidth
1	"Stir fried lettuce recipe"	"Stir fry lettuce recipe"	1		
2	"Sweetened coffee"	"Swedish coffee"		1	1
3	"Sail away song"	"Sell away some"		1	
4	"Let's catch up"	"Let's ketchup"	1	1	1

So far, most of the error analyses are done manually, in a Jupyter notebook or a spreadsheet (like the example above). There are emerging MLOps tools that are making this process easier.

Iterative process of error analysis: Proposing tags

The goal is to come up with a few categories where you could productively improve the algorithm.



Visual inspection:

- Specific class labels (scratch, dent, etc.) ✓
- Image properties (blurry, dark background, light background, reflection, ...)
- Other meta-data: phone model, factory

Product recommendations:

- User demographics
- Product features/category

Useful metrics for each tag

As you're going through the tags, here are a few useful metrics to look at,

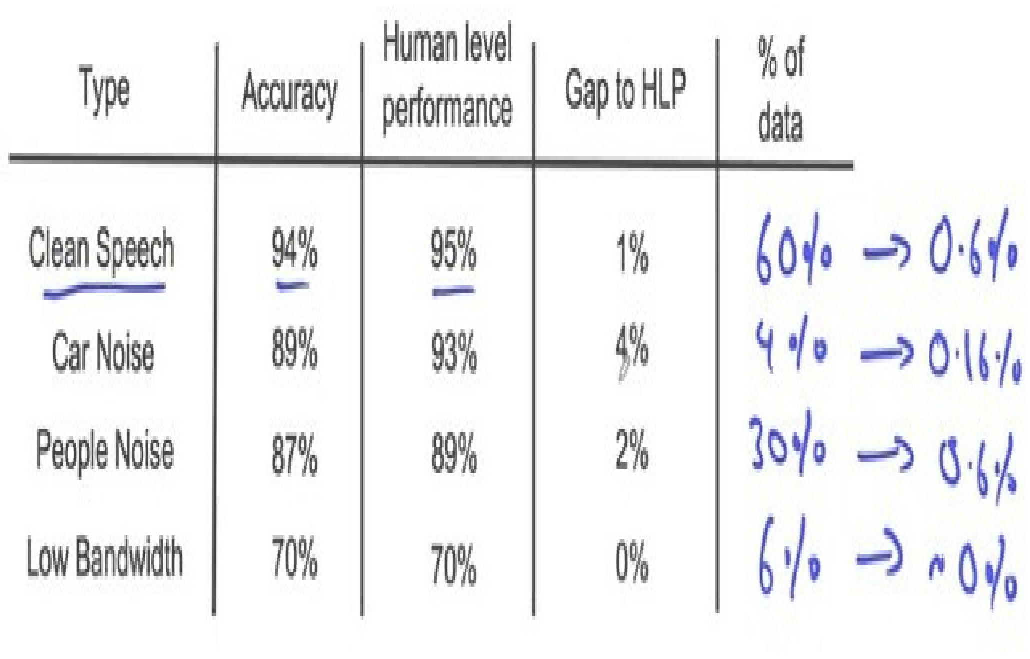
- What fractions of errors have that tag?
- Of all the data with that tag, what fraction is misclassified?

- What fraction of data have that tag?
- How much room for improvement is there on data with that tag?

Prioritizing what to work on

In addition to comparing different tags' performance to that of the baseline, one other useful metric to look at is the **percentage of data with that tag**.

In the example below, the *percentage of data* for each tag tells us that we are better work more on improving performance Clean Speech and People Noise whereas solely looking at the Gap to HLP would suggest we should work on Car Noise tag.



Type	Accuracy	Human level performance	Gap to HLP	% of data
Clean Speech	94%	95%	1%	60% → 0.6%
Car Noise	89%	93%	4%	4% → 0.16%
People Noise	87%	89%	2%	30% → 0.6%
Low Bandwidth	70%	70%	0%	6% → ~0%

Prioritizing what to work on

Decide on the most important categories to work on based on:

- How much room for improvement there is.
- How frequently that category appears.
- How easy it is to improve accuracy in that category.
- How important it is to improve in that category.

There's no mathematical formula to tell you what to work on, but by looking at these factors, you should be able to make more fruitful decisions.

Adding/improving data for specific categories

Once you decided that there's a category (or a few categories) to improve the average performance, consider adding data or improving the quality of the data for that category.

For categories you want to prioritize:

- Collect more data
- Use data augmentation to get more data
- Improve label accuracy/data quality

Going after improving data quality is generally time-consuming and expensive. By carrying out an analysis (like the above), you know exactly what type of data you need to collect. It makes the efforts more focused and efficient.

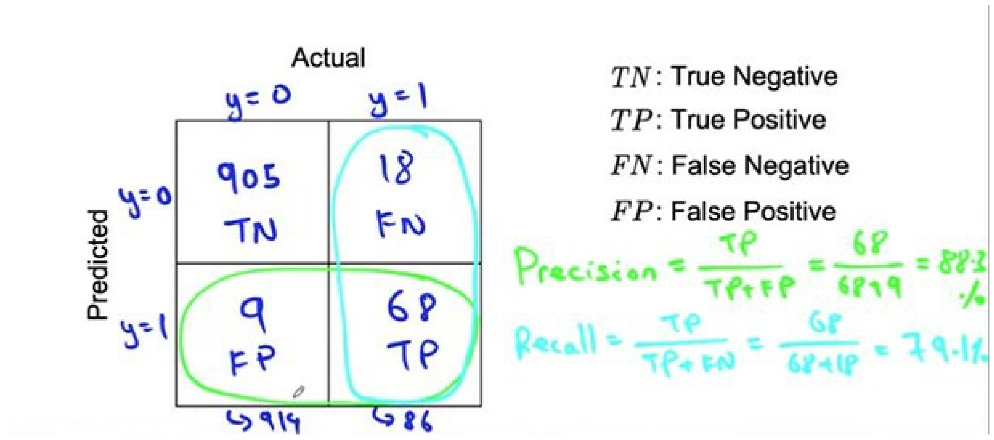
Skewed datasets

Datasets, where the ratio of positive and negative examples is very far from 50-50, are called **skewed datasets**.

Examples of skewed datasets:

- **Manufacturing**
    - 99.7% no defect, %0.3 defect ⇒ just print(0) will achieve 99.7% accuracy.
- **Medical diagnosis**
    - 99% of patients don't have a disease.
- **Speech recognition**
    - In wake word detection, 96.7% of the time wake word doesn't occur.

In skewed datasets, using accuracy is not a good idea because just `print(0)` can get very high accuracy. Instead, it's more useful to build a **confusion matrix**.



With the confusion matrix, if your algorithm outputs 0 all the time, it won't do good on *recall*.

F1 score

Sometimes you have a model with better recall and a different model with better precision. How do you compare these two models? There's a common of doing that using **F1 score**.

The intuition behind the F1 score is that you want an algorithm to do well on both precision and recall, and if it does worse on either of them, that's pretty bad. F1 score is a way of combining precision and recall that emphasizes whichever of P or R is worse.

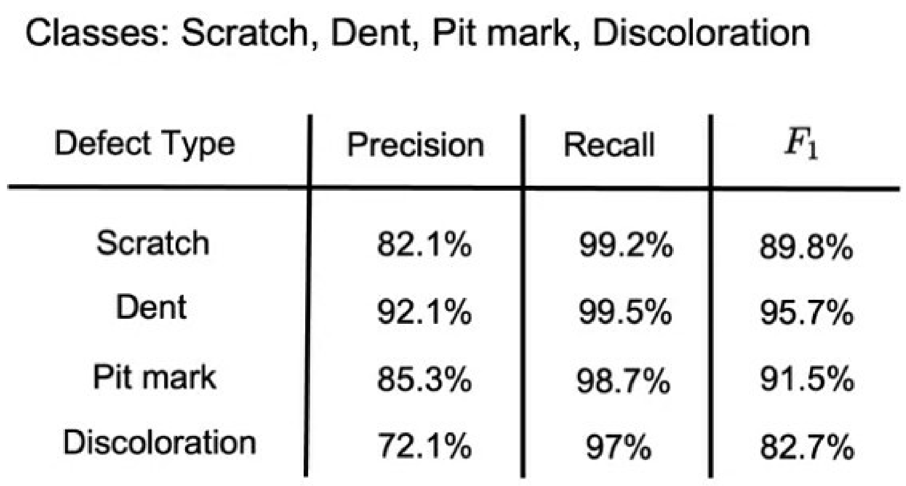
$$F1 = \frac{2}{\frac{1}{P} + \frac{1}{R}}$$

In mathematics, the above formula is technically called a **harmonic mean** between precision and recall, which is like taking an average but placing more emphasis on whichever is the lower number.

Note: F1 score is just one way of comparing models based on precision and recall. There are applications where precision and recall weighting are different.

Multi-class metrics

Let's say you're detecting defects in smartphones, you may want to detect different types of defects.


Classes: Scratch, Dent, Pit mark, Discoloration

Defect Type	Precision	Recall	F_1
Scratch	82.1%	99.2%	89.8%
Dent	92.1%	99.5%	95.7%
Pit mark	85.3%	98.7%	91.5%
Discoloration	72.1%	97%	82.7%

Performance Auditing

Even when your algorithm is doing good on F1 or accuracy, it’s often worth one last performance audit before you push it to production.

Auditing framework

Check for accuracy, fairness/bias, and other problems:

1. Brainstorm the ways the system might go wrong
   - Performance on subsets of data (e.g. ethnicity, gender)
   - How common are certain errors (e.g. FP, FN)
   - Performance in rare classes.
2. Establish metrics to assess the performance against these issues on appropriate slices of data.
3. Get business/product owner buy-in.

Speech recognition example

1. Brainstorm the ways the system might go wrong.
   - Accuracy on different genders and ethnicities
   - Accuracy on different devices
   - Prevalence of rude mis-transcriptions
2. Establish metrics to assess the performance against these issues on appropriate slices of data.
   - Mean accuracy for different genders and major accents.
   - Mean accuracy on different devices.
   - Check for the prevalence of offensive words in the output.

Note: The ways that a system could go wrong tends to be very ***problem-dependent***.

Note: Rather than just one person trying to brainstorm what could go wrong, for high stakes applications if you could have a team (or external advisers) could help you brainstorm things that you want to watch out for and reduce the risk of the project not working.