ML_day9

◇ Types of data :

A dataset could contain many kinds of datatypes, including but certainly not limited to:

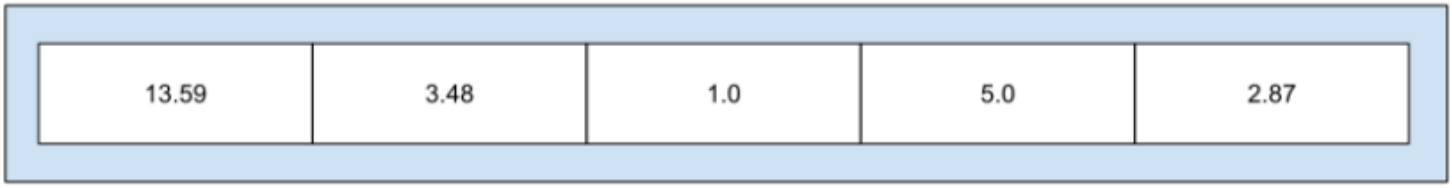
• numerical data, which is covered in a seperate unit
• categorical data, which is covered in a seperate unit
• human language, including individual words and sentences, all the way up to entire text documents
• multimedia (such as images, videos, and audio files)
• outputs from other ML systems
• Embedding vectors , which are covered in a later unit

◇ Quantity of data :

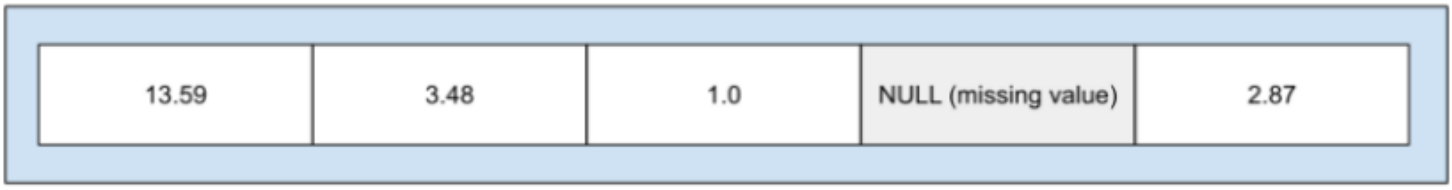
• Models trained on large datasets with few "features" generally outperform models trained on small datasets with a lot of features.
• It's possible to get good results from a small dataset if you are adapting an existing model already trained on large quantities of data from the same schema.

◇ Complete vs. incomplete examples :

• In a perfect world, each example is **complete**; that is, each example contains a value for each feature.



• Unfortunately, real-world examples are often **incomplete**, meaning that at least one feature value is missing.



• Don't train a model on incomplete examples. Instead, fix or eliminate incomplete examples by doing one of the following :

• Delete incomplete examples.
• Impute missing values; that is, convert the incomplete example to a complete example by

providing well-reasoned guesses for the missing values.

◇ Labels :

• Direct labels, which are labels identical to the prediction your model is trying to make.
• For example, a column named bicycle_owner would be a direct label for a binary classification model that predicts whether or not a person owns a bicycle.

• **Proxy labels**, which are labels that are similar—but not identical—to the prediction your model is trying to make.
• For example, a person subscribing to Bicycle Bizarre magazine probably—but not definitely—owns a bicycle.

• Direct labels are generally better than proxy labels. If your dataset provides a possible direct label, you should probably use it. Oftentimes though, direct labels aren't available.

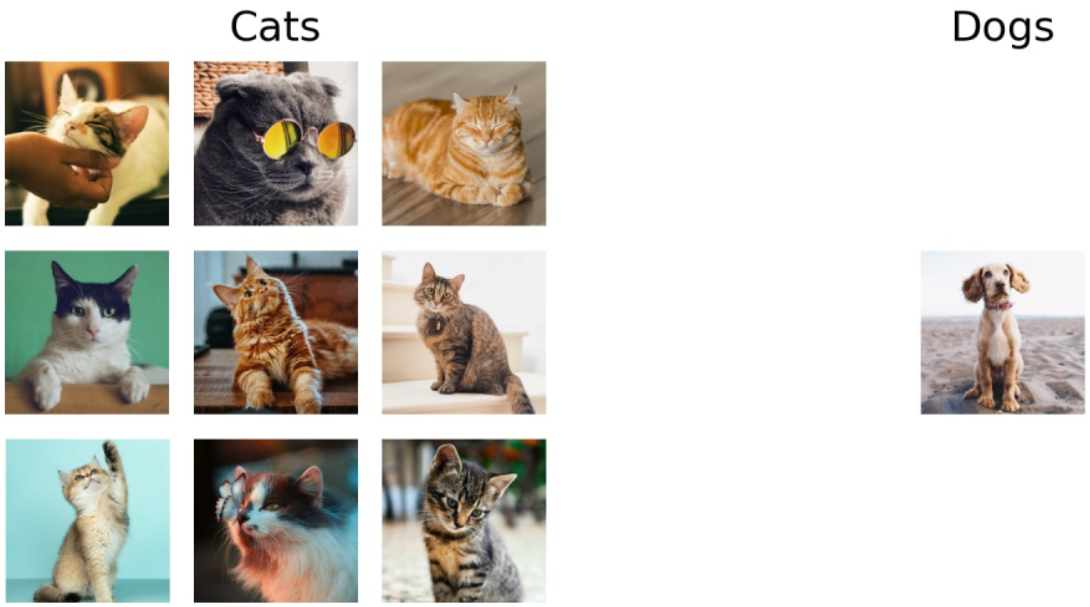
◇ Class-balanced datasets versus class-imbalanced datasets :

• Consider a dataset containing a categorical label whose value is either the positive class or the negative class. In a class-balanced-dataset, the number of positive classes and negative classes is about equal. For example, a dataset containing 235 positive classes and 247 negative classes is a balanced dataset.
• In a class-imbalanced dataset, one label is considerably more common than the other. In the real world, class-imbalanced datasets are far more common than class-balanced datasets. For example, in a dataset of credit card transactions, fraudulent purchases might make up less than 0.1% of the examples. Similarly, in a medical diagnosis dataset, the number of patients with a rare virus might be less than 0.01% of the total examples. In a class-imbalanced dataset:

◇ The *more* common label is called the majority class.
◇ The *less* common label is called the minority class.

• Training aims to create a model that successfully distinguishes the positive class from the negative class. To do so, batches need a sufficient number of *both* positive classes and negative classes. That's not a problem when training on a mildly class-imbalanced dataset since even small batches typically contain sufficient examples of both the positive class and the negative class. However, a severely class-imbalanced dataset might not contain enough minority class examples for proper training.

◇ For example :



• If the batch size is 20, most batches won't contain any examples of the minority class. If the batch size is 100, each batch will contain an average of only one minority class example, which is insufficient for proper training. Even a much larger batch size will still yield such an imbalanced proportion that the model might not train properly.

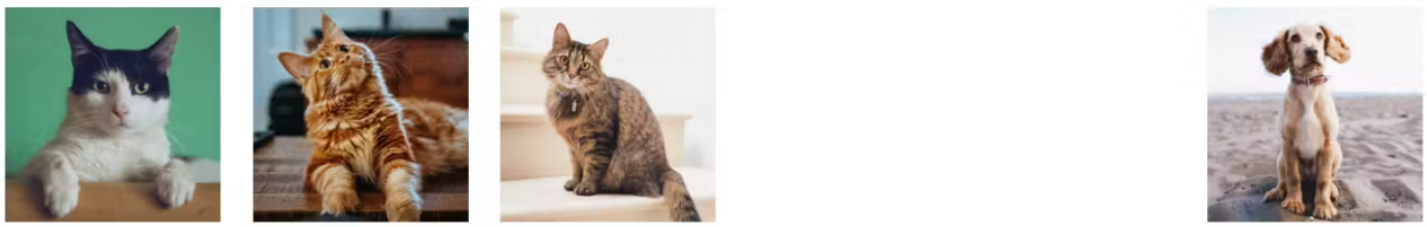
◇ Training a class-imbalanced dataset :

During training, a model should learn two things:

• What each class looks like; that is, what feature values correspond to what class?
• How common each class is; that is, what is the relative distribution of the classes?

• Standard training conflates these two goals. In contrast, the following two-step technique called **downsampling and upweighting the majority class** separates these two goals, enabling the model to achieve *both* goals.

• Downsampling means training on a disproportionately low percentage of majority class examples. That is, you artificially force a class-imbalanced dataset to become somewhat more balanced by omitting many of the majority class examples from training. Downsampling greatly increases the probability that each batch contains enough examples of the minority class to train the model properly and efficiently.



• Here instead of 9 cat photos we are giving 3 cats and 1 dog .

◇ Upweight the downsampled class :

• Downsampling introduces a "prediction bias" by showing the model an artificial world where the classes are more balanced than in the real world. To correct this bias, you must "upweight" the majority classes by the factor to which you downsampled.

• For example, we downsampled the majority class by a factor of 25, so we must upweight the majority class by a factor of 25. That is, when the model mistakenly predicts the majority class, treat the loss as if it were 25 errors (multiply the regular loss by 25).

◇ Benefits of this technique :

"Downsampling" and "upweighting" the majority class brings the following benefits:

• **Better model:** The resultant model "knows" both of the following:

◇ The connection between features and labels
◇ The true distribution of the classes

• **Faster convergence:** During training, the model sees the minority class more often, which helps the model converge faster.