Reading: More about imbalanced datasets

Imbalanced Datasets

As you may have noticed as you've gone through this program, many raw datasets that you will encounter require varying levels of work to get them in a place where they are ready for modeling. There might be missing values, or the variables might not be in the exact format that you need. You perform an exploratory data analysis to get a good understanding of the data. When working with classification models, however, there is something else to consider before moving forward: class balance.

For categorical variables, the different possible values that each can take are known as classes. This is true for both predictor variables and target variables. If you were trying to classify the weather on a given day as rainy or sunny, these would be considered two classes. The number of classes is equal to the number of unique values in the variable.

The number of occurrences of each class in the target variable is known as the class distribution. When predicting a categorical target, problems can arise when the class distribution is highly imbalanced. If there are not enough instances of certain outcomes, the resulting model might not be very good at predicting that class.

This is where the process of class balancing comes in, a process that allows you to manipulate the dataset, or the model fitting process, in a way that the class imbalance that exists doesn't affect the performance of the resulting model. In this reading, you will explore more about imbalanced datasets, the problems that can occur when working with them, and some methods of adjusting the class distribution to minimize the imbalance.

Situations of Imbalance

Classification is a very broad field, with many applications across different industries. Some business needs, however, require a model to be good at classifying things that occur relatively rarely in the data. These types of problems have several names that are used commonly, including rare event prediction, extreme event prediction, and severe class imbalance. However, they all refer to the same thing: at least one of the classes in the target variable occurs much less frequently than another.

Consider this example. You are tasked with creating a model that will classify emails that are sent to the company either as "spam" or "not spam." The company receives thousands and thousands of emails daily, not to mention all the emails they've received in the past. However, the number of examples of spam is very small. For the sake of this example, say that 10 emails per day are identified as spam manually.

All the emails are collected into a dataset to train the model, with each example labeled as spam or not spam. The problem is that the dataset contains many, many more examples of not spam than spam. In this case, spam is known as the minority class and not spam is known as the majority class.

This doesn't have the makings of a very good model. With so few examples of spam relative to examples of not spam, the model can have difficulty detecting the minority class, resulting in its being biased toward the majority class or possibly never predicting the minority class at all.

Balancing a Dataset

Class balancing refers to the process of changing the data by altering the number of samples in order to make the ratios of classes in the target variable less asymmetrical. It is a large field of study on its own, and there are several methods that allow you to balance the classes while maintaining the integrity of the data. Here, you'll learn about some of the most common methods that can be used to create a better model.

There are two general strategies to balance a dataset, and the method that is better to use generally is decided by how much data you have in the first place.

Downsampling

Downsampling is the process of making the minority class represent a larger share of the whole dataset simply by removing observations from the majority class. It is mostly used with datasets that are large. But how large is large enough to consider downsampling? Tens of thousands is a good rule of thumb, but ultimately this needs to be validated by checking that model performance doesn't deteriorate as you train with less data.

One way to downsample data is by selecting some observations randomly from the majority class and removing them from the dataset. There are some more technical, mathematically based methods, but random removal works very well in most cases.

Upsampling

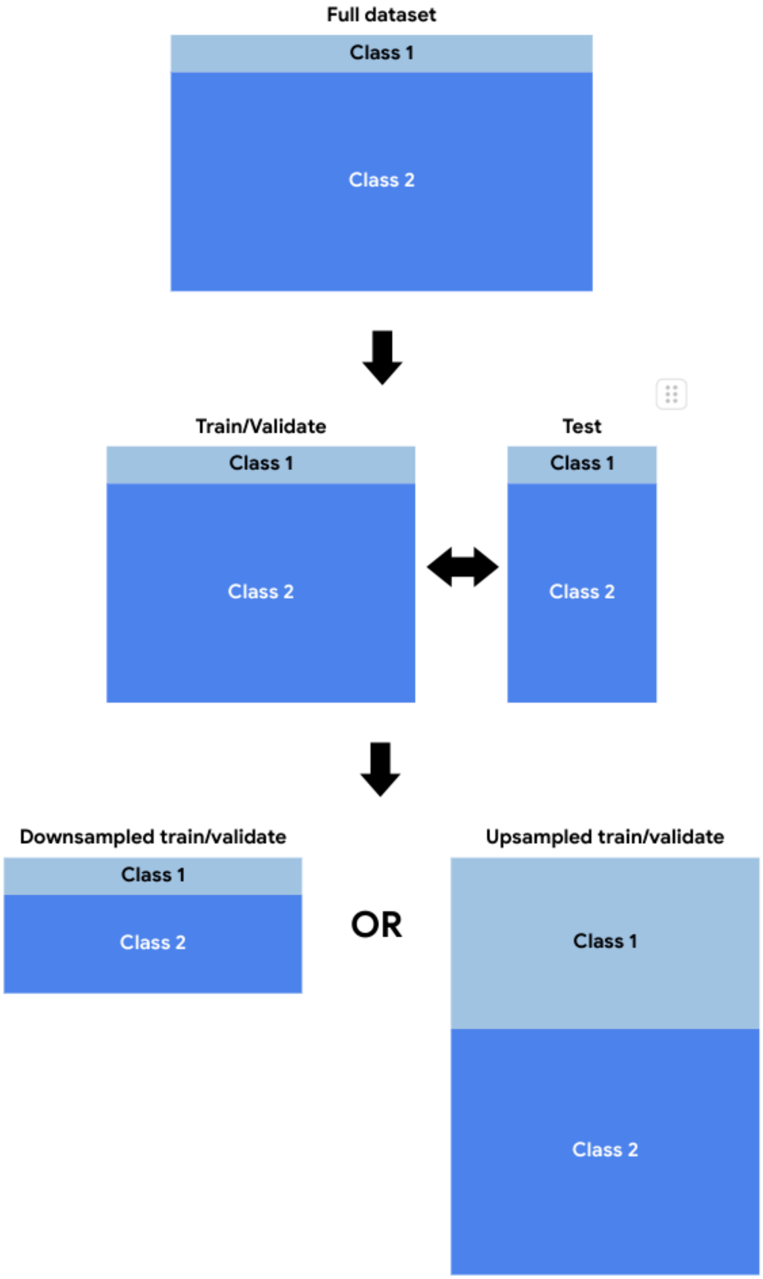
Upsampling is basically the opposite of downsampling, and is done when the dataset doesn't have a very large number of observations in the first place. Instead of removing observations from the majority class, you increase the number of observations in the minority class.

There are a couple of ways to go about this. The first and easiest method is to duplicate samples of the minority class. Depending on how many such observations you have compared to the majority class, you might have to duplicate each sample several times over.

Another way is to create synthetic, unique observations of the minority class. On the surface, there seems to be something wrong about editing the dataset like this, but if the goal is simply to train a better-performing model, it can be a valid and useful technique. You can generate these synthetic observations from the observations that currently exist. For example, you can average two points of the minority class and add the result to the dataset as a sample of the minority class. This can even be done algorithmically using publicly available Python packages.

How to do it

In both cases, upsampling and downsampling, it is important to leave a partition of test data that is unaltered by the sampling adjustment. You do this because you need to understand how well your model predicts on the actual class distribution observed in the world that your data represents. In the case of the spam detector example, it's great if your model can score well on resampled data that is 80% not spam and 20% spam, but you need to know how it will work when deployed in the real world, where spam emails are much less frequent. This is why the test holdout data is not rebalanced.



Consequences

Manipulating the class distribution of your data doesn't come without consequences. The first consequence is the risk of your model predicting the minority class more than it should. By class rebalancing to get your model to recognize the minority class, you might build a model that over-recognizes that class. That happens because, in training, it learned a data distribution that is not what it will be in the real world.

Changing the class distribution affects the underlying class probabilities learned by the model. Consider, for example, how the Naive Bayes algorithm works. To calculate the probability of a class, given the features, it uses the background probability of a class in the data.

$$P(A|B) = \frac{P(B|A).P(A)}{P(B)}$$

In the numerator, *P(A)* (the probability of class *A*) depends on class probabilities encountered in the data. If the data has been enriched with a particular class, then this probability will not be reflective of meaningful real-life patterns, because it's based on altered data.

When to do it

Class rebalancing should be reserved for situations where other alternatives have been exhausted and you still are not achieving satisfactory model results. Some guiding questions include:

- How severe is the imbalance? A moderate (< 20%) imbalance may not require any rebalancing. An extreme imbalance (< 1%) would be a more likely candidate.
- Have you already tried training a model using the true distribution? If the model doesn't fit well due to very few samples in the minority class, then it could be worth rebalancing, but you won't know unless you first try without rebalancing.
- Do you need to use the model's predicted class probabilities in a downstream process? If all you need is a class assignment, class rebalancing can be a very useful tool, but if you need to use your model's output class probabilities in another downstream model or decision, then rebalancing can be a problem because it changes the underlying probabilities in the source data.

Key Takeaways

- Imbalanced datasets can be a problem when working on classification problems
- Class imbalance isn't always a problem. The likelihood and severity of it negatively affecting model performance generally increase as the degree of the imbalance increases.
- Downsampling involves removing some observations from the majority class, making it so they make up a smaller percent of the dataset than before.
- Upsampling involves taking observations from the minority class and either adding copies of those observations to the dataset or generating new observations to add to the dataset.

Resources for more information

- Google Developers: Further reading about class imbalance
- imbalanced-learn: Introduction to imbalanced-learn, a library with tools to help with unbalanced datasets. Designed to work with scikit-learn.
- RandomOverSampler: imbalanced-learn documentation for a tool used to randomly upsample data
- Upsampling methods: imbalanced-learn documentation for various upsampling methods
- Downsampling methods: imbalanced-learn documentation for various downsampling methods