1. Combining timeseries data with machine learning

In the final lesson of this chapter, we'll discuss the interaction between machine learning and timeseries data, and introduce why they're worth thinking about in tandem.

2. Getting to know our data

First, let's give a quick overview of the data we'll be using. They're both freely available online, and come from the excellent website Kaggle-dot-com.

3. The Heartbeat Acoustic Data

Audio is a very common kind of timeseries data. Audio tends to have a very high sampling frequency (often above 20,000 samples per second!). Our first dataset is audio data recorded from the hearts of medical patients. A subset of these patients have heart abnormalities. Can we use only this heartbeat data to detect which subjects have abnormalities?

4. Loading auditory data

Audio data is often stored in "wav" files. We can list all of these files using the "glob" function. It lists files that match a given pattern. Each of these files contains the auditory data for one heartbeat session, as well as the sampling rate for that data.

5. Reading in auditory data

We'll use a library called "librosa" to read in the audio dataset. Librosa has functions for extracting features, visualizations, and analysis for auditory data. We can import the data using the "load" function. The data is stored in audio and the sampling frequency is stored in sfreq. Note that the sampling frequency here is 2205, which means 2205 samples are recorded per second.

6. Inferring time from samples

Using only the sampling frequency, we can infer the timepoint of each datapoint in our audio file, relative to the start of the file.

7. Creating a time array (I)

Now we'll create an array of timestamps for our data. To do so, you have two options. The first is to generate a range of indices from zero to the number of datapoints in your audio file, divide each index by the sampling frequency, and you have a timepoint for each data point.

8. Creating a time array (II)

The second option is to calculate the final timepoint of your audio data using a similar method. Then, use the linspace function to generate evenly-spaced numbers between 0 and the final timepoint. In either case, you should have an array of numbers of the same length as your audio data.

9. The New York Stock Exchange dataset

Next, we'll explore data from the New York Stock Exchange. It runs over a much longer timespan than our audio data, and has a sampling frequency on the order of one sample per day (compared with 2,205 samples per second with the audio data). Our goal is to predict the stock value of a company using historical data from the market. As we are predicting a continuous output value, this is a regression problem.

10. Looking at the data

Let's take a look at the raw data. Each row is a sample for a given day and company. It seems that the dates go back all the way to 2010.

11. Timeseries with Pandas DataFrames

It is useful to investigate the "type" of data in each column. Numpy or Pandas may treat an array of data in special ways depending on its type. We can print the type of each column by looking at the

dot-dtypes attribute. Here we see that the type of each column is "object", which is a generic data type.

12. Converting a column to a time series

Since we know one column is actually a list of dates, let's change the column type to "datetime" using the to_datetime function. This will help us perform visualization and analysis later on.