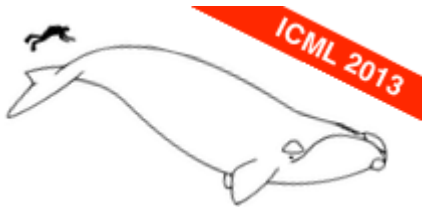

The ICML 2013 Whale Challenge - Right Whale Redux

Develop recognition solutions to detect and classify right whales for BIG data mining and exploration studies

$500 · 129 teams · 5 years ago

Christopher Hefele
2nd place

The Leakage (and how it was fixed)

posted in The ICML 2013 Whale Challenge - Right Whale Redux 5 years ago

29

As you know, this contest was restarted because some data leakage was found in the first release of the dataset. Now that the contest is over, I'd like to talk a bit about that leakage; I think the technical details are interesting, and they might make for an interesting case study.

I found that it was possible to get an 0.9973 AUC using only leakage, *without* reading the audio files we were given. For a brief moment, that put my team ("Leaky Larry") on top of the leaderboard and ahead of legitimate teams that created the top algorithms in the first Whale Detection Challenge.

Leakage can, of course, undermine the credibility of the leaderboard scores. In the first week of the contest, I gave Kaggle the details about the leakage I found. Cornell / Marineexplore then promptly revised the dataset, and then the competition was restarted. The revision fixed most of the issue, and the algorithm that got 0.9973 on the original dataset (using only leakage) could only achieve a 0.59 AUC on the revised dataset.

The leakage came from three sources:

1. The distribution of file lengths (in bytes)
2. The timestamp embedded in the audio clip filename (focusing mostly on the millisecond field)
3. The chronological order of the clips (ordered using the full timestamp)

File Sizes

After downloading the data and typing a simple "ls -lS", I noticed that a large number of audio clip files had both the same size and the same label (i.e. whale vs no-whale). Interestingly, flles with the same size were not duplicates (their md5sums differed). This seemed fishy.

To investigate further, I made histograms of the file sizes, broken down by label. Audio files with whale upcalls turned out to have a very specific set of file sizes (see this "comb-like" histogram where some files are multiples of 148 bytes plus a constant). In contrast, files without whale upcalls had sizes that looked much more evenly distributed (see this histogram). Thus, certain file sizes provided a strong indication that a clip had a whale upcall in it. Also, files with whale upcalls were larger, on average.

Millisecond timestamps

Another anomaly was related to the timestamps embedded in the filenames. If a whale upcall was not in a file, the millisecond field in the timestamp was almost always a multiple of 10 ms (see this). However, if a whale upcall was in a file, the millisecond field seemed evenly distributed in time - i.e. multiples of 1 ms (see this). Thus, a zero in the last digit of the millisecond field was strongly predictive; using a simple test for zero as a binary feature yielded a 0.945 AUC by itself.

Next, additional histograms of the millisecond timestamps showed that the audio clips without whale upcalls were more likely to start in the first half of a given second (see this). Clips with whale upcalls were more evenly distributed across time (see this).

Putting these observations together, it seems like the audio clips with upcalls were processed in a different way than those without upcalls. As a competitor, though, it's impossible to tell what the true

root causes of these differences were, or if they could be useful.

Clip order

in the first Whale Detection challenge, the (chronological) ordering of the clips contained information. The same was true in this contest. A moving average of clip labels in the training set showed a familiar pattern: minutes or hours of high whale-call activity, followed by equally lengthy lulls (see this). A moving average could be used to capture some of this serial correlation in the leaked test clip labels, providing a "temporally local" probability of a whale upcall.

Features

To take advantage of all these observations, I created the following features:

- File size in bytes (as an integer)
- File size in bytes (as a categorical variable).
- Timestamp_milliseconds (as an integer)
- 0 if timestamp_milliseconds was a multiple of 10, 1 otherwise
- 0 if timestamp_milliseconds was a multiple of 10, moving average of the above 0/1 feature otherwise

A logistic regression using these simple features yielded 0.9973 AUC on the leaderboard. I used the 'glmnet' package in R for logistic regression, and python to create the features.

Fixes

Cornell corrected these anomalies by:

- Capping the file size so that most files were exactly 2 seconds long (8088 bytes).
- Reducing the millisecond field in the filename timestamp from 3 digits down to 1 (though what that 1 digit represents is unclear).

I think it's easy to see how these slight anomalies could go unnoticed, though some turned out to be unexpectedly effective predictors. Even if competitors didn't notice these these anomalies, it's possible

that some algorithms might have silently picked up on them if they weren't corrected.

Like software bugs, leakage seems hard to avoid entirely. I applaud Kaggle & Cornell for swiftly fixing these issues once they were found.

Options

Comments (2)

Sort by Hotness

Please sign in to leave a comment.

Anil Thomas • (15th in this Competition) • 5 years ago • Options

2

Chris, thanks for sharing. This is more fascinating than the contest itself. Who knew the 'ls' command is a powerful data exploration tool!

Rhodium Beng • 2 months ago • Options

0

Fascinating story on data leakage. Thank you for sharing.