Contest: Fishing For Fishermen Marathon
Name: Wladimir Leite
TC Handle: wleite
Email: wladimir.leite@gmail.com

Approaches considered

When I read the problem for the first time I decided to start with my standard random forest implementation, using only AIS messages (no static data), then add static/trip data, and finally external data.

Initially I considered using a library to parse the messages, but realized that parsing myself was straightforward, since although specification is somewhat complex, the set of message types/subtypes we have to deal was limited.

Approach ultimately chosen

It uses the classic training and testing framework, using AIS messages and vessel information extracted from static data. The next section describes the relevant aspects of my solution.

Steps to approach ultimately chosen, including references to specific lines of your code

Parsing AIS messages: I followed the provided specification and implemented a simple parser that deals only with the message types present in provided data (1-3, 18, 19 and 27), and potentially useful fields. I noted that there are malformed messages missing the first exclamation mark (before “AIVDM”). After checking/fixing this condition, all messages passed the checksum verification.

Parsing Static Data: I used the same code that parsed AIS messages, with specific treatment for different message types (5 and 24). Although I tried many different things here, in the end I used only static data: ship and dimensions type, which were loaded to a map, that used the MMSI as key.

Duplicated Training Entries: Duplicated entries were removed. Fields checked to consider two entries as duplicated were: latitude, longitude, message type, MMSI, time, timestamp, speed over ground, repeat count and fishing status.

Unknown Training Entries: I considered discarding them, or using multi-class (instead of binary) classification, but decided to use a simpler solution: just add two identical entries, one marked as “fishing” and other “not fishing”. This was trivial to code, and I guess it gave the random forest enough information to mark these entries as ambiguous.

Features Used: By far what I spent most time was trying all different features that came to my mind. I used different categories of features: standard, averaged by MMSI (took from all entries with the same MMSI), mapped categories (instead of using raw values/types, a map based on average fishing percentage was built from training data), static (from static data, mapped by MMSI), and time based (comparing sequential entries from the same vessel). In the end I could find any useful time based features, after testing many of them (e.g. distance from the previous location, speed change, course change).

Final list of features used, grouped by category:

- Standard:
  - Latitude;
  - Longitude;
  - Maneuver Indicator;
  - Speed Over Ground;
  - Channel Code;
  - Month (from entry’s time);
  - Day of the Week (1 to 7, from entry’s time);
  - Message Type;
  - Repeat Count;

- Averaged by MMSI:
  - Latitude;
  - Repeat Count;
- Standard, Category Mapped to Average Fishing Value:
  - Navigation Status;
- Static:
  - Dimension to Bow;
  - Dimension to Starboard;
  - Dimension to Stern
  - Dimension to Port;
- Static, Category Mapped to Average Fishing Value:
  - Ship type.

Training/Predicting: Just built 64 binary classification trees and used the simple average result as the prediction.

Open source resources and tools used, including URLs and references to specific lines of your code

I spent quite some time trying to find public available data sources that could be used to improve my predictions.

The most promising candidates I found were **water temperature** for a given month and location (latitude and longitude on 1/4-degree grid) and **water salinity** (similar, but 1-degree resolution). After implementing a way to parse and use these files, and didn't manage to see any improvement in the predictions, so I decided not to use them, as it added extra code and extra files to deal with.

Advantages and disadvantages of the approach chosen

It is a straightforward approach, and simple to understand/improve. Not using ML libraries prevent for trying more sophisticated classification methods.

Comments on libraries

I decided not to use any libraries.

Comments on open source resources used

My final submission did not use any open source resource.

Potential improvements to my algorithm

Although I spent a lot of time working on feature engineering, I ended up discarding a lot of potential features that could add some information, because they didn't help at all in my local tests with training data portions, neither improved my provisional score. So there is room for improvement here.

Of course using external data may also help, but I am not sure if there are public data sources available that would be useful in this case.