2d Lt Nico De Ros

Practical Machine Learning Final Project Report

11 June 2024

Hooked or High Seas:

Neural Networks for Fishing Status Classification

Introduction

What happens in international waters stays in international waters, right? Well, when it comes to illegal fishing, Global Fishing Watch is making an effort to bring this malpractice to light. Illegal fishing in international waters presents unique challenges due to the absence of a single governing authority with jurisdiction over these areas. International waters, also known as the high seas, lie beyond any country's exclusive economic zone (EEZ) and are considered part of the global commons. The lack of clear jurisdiction and enforcement mechanisms makes international waters vulnerable to illegal fishing activities, including Illegal, Unreported, and Unregulated (IUU) fishing. Vessels operating in these areas may exploit regulatory gaps, evade oversight, and engage in practices that are prohibited in national waters, such as overfishing, shark finning, and the use of destructive fishing gear.

To combat IUU fishing, Global Fishing Watch has created a database specifically to be used for detecting fishing events as well as gear type by analyzing AIS vessel tracks. An artificial neural network (ANN) is a good choice to analyze this database as its vast size and nuanced features would make it otherwise impossible for an analyst to make sense of by hand or with less powerful computational methods. While the data is public, there are no ANN models that have been publicly published to act as a baseline for the ANN proposed in this research. Instead, the baseline that the ANN proposed in this research will compare to is a randomly

generated ANN with fixed hyperparameters. The created ANN will then use hyperparameter tuning to see how much more improved it can become. Following Chollet's 7 Step Process, the rest of this paper will provide a background that discusses the data and its preprocessing, methodology for incrementally developing the ANN, results and a comparison to the baseline, and concluding thoughts of the entire process.

Background

As mentioned in the introduction, the data provided comes from globalfishingwatch.org, titled "Anonymized AIS Training Data". This dataset is public and contains over 14 million unique records that were last updated in 2020. Each record has a value for each of its 9 features which are briefly described here:

mmsi: the unique, 13-digit MMSI of a ship,

timestamp: UNIX timestamp (measured in seconds) of the observations' recording,

distance_from_shore: Distance from shore in meters,

distance_from_port: Distance from port in meters,

speed: Vessel speed in knots,

course: Vessel course in degrees,

lat: Latitude in decimal degrees,

lon: Longitude in decimal degrees, and

source: Data was prepared by GFW, Dalhousie, and a crowd sourcing campaign. False positives are marked as false_positives.

Each record also has a measurement for its response, *is_fishing*. This numerically encoded response can either be a -1 (no recorded data), a 0 (not fishing when observed), a 1 (fishing when

observed), or it can be a decimal value between 0 and 1 if it was observed by multiple agents (the average of the reports).

The measure of success that will be used to determine how well a model performs is accuracy. Accuracy in a model refers to the proportion of correctly classified instances out of the total instances evaluated. It is a common metric used to assess the performance of classification models, which are used to predict categorical outcomes based on input features. However, accuracy alone may not always provide a complete picture of a model's performance, so additional measures of success such as recall and precision will also be used to understand if there is improvement in more than a single metric.

While Global Fishing Watch did an excellent job of ensuring that their datasets only contain complete observations, there was still a hefty amount of preprocessing that needed to be completed. First, their data was split into different comma separated value (csv) files by the boat type being observed. To make preprocessing easier, and to make a final model that would be more generalizable, all the data was loaded into a Python environment and aggregated into a single dataset. After this, it was deemed prudent to remove all the data that did not have a measured response (rows where *is_fishing* was -1). Of the 14.6 million rows of data, this single operation reduced the number of rows to just over 300 thousand. Of the 300 thousand observations, around 8,000 had a response that was between 0 and 1. As a personal choice, it was deemed prudent to change all these values to 1. This eliminated class imbalance between never observed fishing, always observed fishing, and sometimes observed fishing, and turned our problem into a binary classification problem of, “Is the identified boat fishing or not?”

After this step, it was important to check for missing values, zeros, and extreme or unlikely values for each of the features. All other columns that had a 0 in any of their rows were

determined to be logical and thus correctly inputted as a zero. For example, the *speed* column has more than 130 thousand rows with a zero in them. This makes sense because, for example, a boat can have a speed of 0 (moored at a dock or anchored at sea), and a course can be 0 (heading of true North). Now a closer look should be taken at the extreme values that were noticed.

To do this, the columns will be analyzed one at a time. Starting with MMSI, extreme values do not provide useful information when dealing with identifying numbers. Nothing was observed to be longer than 13 digits, so there is no reason to assume that any of these identifiers were recorded incorrectly. For *timestamp,* there were a variety of UNIX timestamps covering multiple years' worth of data collection. UNIX is represented in seconds, so to cover years, you would have some numbers be a hundred billion more or less than another number. All numbers were between 1.3 trillion and 1.5 trillion, so no issues were determined to exist in this variable either.

For *distance_from_shore,* some ships were thousands of kilometers out in deep ocean waters while many more were moored at a dock or shoreline. This variance in scale provided a way for extremes to come about. When looking deeper into the issue, the largest distance from shore was 2,854,625 meters (2,854.6 km). Verifying the reality of this point, it turns out that the point that is farthest from shore in any direction is Point Nemo in the Pacific Ocean, which is at least 2,688 kilometers from land in any direction. This means that the value was either recorded incorrectly, or whoever reported it did not measure to the closest point of land. Perhaps they recorded the distance to their own country's shoreline. I decided to keep these values as Global Fishing Watch does not provide a standard method of measuring this distance, but perhaps the final model could be more accurate if complete case analysis was done on any row with a value above 2,688,000 in its column.

The feature *distance_from_port* had the same issue as *distance_from_shore*, where some boats were logged and reported to be in a port, so their distance was 0, while some were out in open ocean. However, even worse than the *distance_from_shore* feature, the greatest value here was 3836963.25 meters (3836.9 km)! That's an extra 1000 kilometers from Point Nemo to the next closest port, which is very unrealistic. This suggests that it was not an issue with being recorded incorrectly, but an issue with measurement. However, because the website offers no standardized way to measure distances to ports, it was decided to keep these values as well. Maybe this is the distance to the closest major shipping port in the record taker's home country!

For *speed*, many boats were anchored or moored so their speed was 0. This brought the average down, so if a boat was seen speeding across open water, it looked like an extreme value. Nothing was so fast though that it appeared to be incorrectly recorded. For *course*, the largest value observed was 511. A course should only be able to take on values between 0 and 360, which means that any measurement that is 360 or greater is too large and is an incorrect measurement. Taking count, the number of observations that were greater or equal to 360 was 207 which is coincidentally (or maybe not) also the number of extreme values identified. Complete case analysis was done to remove all these observations from the final dataset.

For latitude, it turned out that most values were around the equator (-15 degrees to 15 degrees). A possible reason for this is that the warmer waters not only promote more fishing vessels to be out on the water, but that there are also more ships out on the water who are willing to create log and submit the reports of other ships and their activity, as opposed to colder waters.

While latitude can take on values from -90 to 90, longitude takes on values from -180 to 180. There were longitude values from a variety of places around the world, but the values that kept popping up as extremes were about 170. This is a valid line of longitude, so a standard

world map was observed to see if any insights could be obtained. It turns out 170 longitude is approximately the middle of the Pacific Ocean, so boats going out this far are likely sparse and distanced from each other. Additionally, anyone who was helping to collect data for this report likely would not have wanted to venture here to check for fishing activity. Thus, all extremes for latitude and longitude are accounted for. Finally for *source*, some of the sources of data collection were marked as 'false positives', the meaning of which is unclear. There were 8000 of these values, so all of them were removed as well, just to ensure that our data had nothing icky going on behind the scenes. While the data may have observation bias, it is otherwise clean!

Now that the data has been thoroughly preprocessed with well over 300 thousand observations remaining, the time has come to split the data into training, validation, and testing sets. Because multiple ships had multiple observations, it was deemed analytically important to split the data based on the MMSI values while trying to observe a 65/17.5/17.5 data split. While the keeping MMSI split by the set, the closest to the desired split that could be obtained was a 65/17/18 split into training, validation, and testing sets respectively. At this point, enough background to the problem has been provided and the ANN can be discussed.

Methodology

The goal of the ANN developed here is to provide a binary classification of whether the observed ship is fishing or not. For binary classification tasks, where the goal is to classify input data into one of two categories, several types of ANNs can be effective. However, one of the most used architectures for binary classification is the Feedforward Neural Network (FNN).

FNNs are well-suited for binary classification tasks due to their simplicity and effectiveness. In a typical FNN architecture, the input data is passed through multiple layers of neurons, with each layer applying a set of weights to the input and passing the result to the next layer. The final output layer usually consists of a single neuron with a sigmoid activation function, which squashes the output to a range between 0 and 1, representing the probability of belonging to one of the two classes. To ensure that all the model architecture was reproducible, a seed of 42 was set in any function and before any use of randomness to guarantee reproducibility. This includes setting a seed before the data split so the training set would always have the same, separate data from the testing and validation sets. All models created use the Adam optimizer, binary cross entropy for loss, and accuracy as the metric.

The initial model developed was a simple one, only created to act as the scaffolding for more complex models to come while still fitting the data shape. This model takes the inputs and applies a weighted sum of 16 neurons in a hidden layer before applying a ReLU activation function to it. This information gets passed to the output layer where another weighted sum is applied before using the sigmoid activation function to produce a final binary classification. If the output is 1, that means the information passed along is identified to belonging to a boat that is fishing, whereas if the output is 0, the information passed corresponds to a boat that is identified as not fishing. This process iterates over 5 epochs with a batch size of 32. After the process is over, the recorded loss and accuracy are graphed and recorded for later analysis.

The second model developed was an overfitting of the first. It also expands upon the architecture of the first model by adding more layers and more neurons per layer. The first hidden layer has 128 neurons, the second has 64, the third has 32, and the last hidden layer has 16. Similar to the initial model, the hidden layers each use a ReLU activation function, the batch

size is 32, and the output layer performs binary classification with a sigmoid activation function. Also scaled up, this model uses 20 epochs to learn the training data. Expectedly, this model performs better than the initial model.

Finally, the generalized model is created using regularization techniques. Early stopping, dropout, L1, and L2 regularization techniques were experimented with to get the best model for this part of the process. However, it proved most effective (in terms of accuracy) to use dropout and early stopping together, leaving out L1 and L2 regularization.

The dropout technique involves randomly setting to zero a proportion of the neurons in a layer during training. This means that the output of those neurons is temporarily ignored during the forward pass, and their weights are not updated during backpropagation. By randomly dropping out neurons during training, dropout prevents the network from relying too heavily on specific neurons or co-adapting to the noise in the training data. This encourages the network to learn more robust and generalizable features, ultimately reducing overfitting and improving the model's ability to generalize to unseen data.

Early stopping works on the assumption that as training progresses, the model will continue to improve on both the training and validation datasets. However, at some point, further training may lead to overfitting, where the model starts to memorize the training data rather than learning generalizable patterns. This is often indicated by a divergence between the model's performance on the training and validation datasets, with the validation performance deteriorating while the training performance continues to improve.

Except for the introduction of early stopping and dropout, the generalized model was set up the same as the overfit model. For each layer, a dropout rate of 0.5 was applied, and early stopping was implemented with a patience of 10 epochs. To accommodate for early stopping, the

number of epochs the model was set to train for was 200. This generalized model's hyperparameters would then be fixed, and this model would become the baseline to compare the tuned model to.

To create the final model, hyperparameter tuning was introduced using two different levels for the optimizer, dropout rate, and number of neurons in each layer. For the optimizer, the levels were using the Adam optimizer and the RMSprop optimizer. For the dropout rate, a rate of 0.5 and 0.4 were used. For the neurons in each layer, the levels were the original number of neurons in their respective layer from the generalized model and half of those neurons. A more academically rigorous approach had previously been attempted, but because of the size of the dataset, memory limits had been quickly reached which prevented a thorough search of the hyperparameter space. This current method, while basic, still provided useful insights into how to tweak the hyperparameters to produce a more useful model. Each model was trained and tested for accuracy, with only the model with the highest accuracy being saved. This final model was compared to the baseline model produced by generalizing a model without hyperparameter tuning, so final conclusions could be drawn.

Results

Before analyzing the results, it is important to ask the question, "Is this model just guessing?" Because the ratio of not fishing to fishing was 1:2, a model that was only guessing would have a 66% accuracy. So long as the model accuracy is greater than 66%, confidence can be had that the model is not guessing. The baseline model was a personally developed FNN that had four hidden layers that utilized dropout and early stopping to produce a generalizable model. As a baseline, this model has the following confusion matrix:

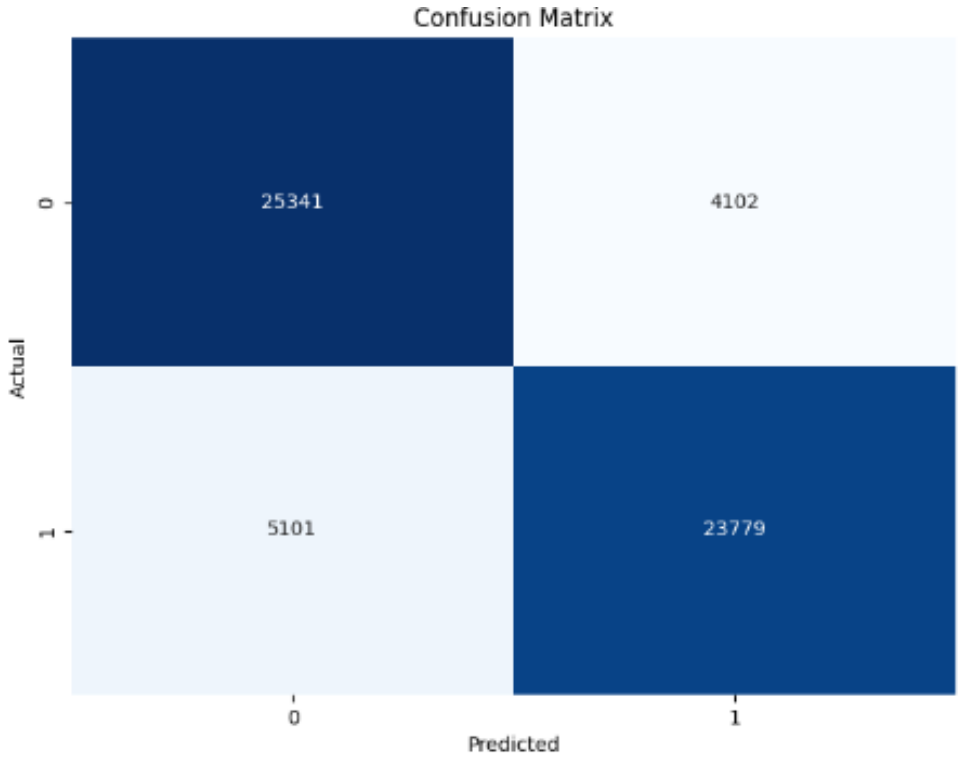

Figure 1: Generalized Model Confusion Matrix

From this confusion matrix, it can be found that the accuracy is 0.8422. This means that the model correctly predicts the class of a ship 84.22% of the time. Additional metrics that can be used to analyze the model's usefulness are precision and recall. Precision is defined as how accurately the model predicts events, so it is desirable for this value to be as high as possible. Recall tells an analyst how accurately the model classifies actual events, so this value should be close to 1 as well. First looking at the precision for fishing ships, it is calculated by using: (23,779) / (23,779 + 4,102) = 0.8529. The recall is: (23,779) / (23,779 + 5,101) = 0.8234. These values are close to 1, so confidence can be had in the baseline models' predictive performance. Additionally, the reported loss of the model was 0.3411. Ideally this would be 0, but this is still low enough to proceed with the belief that this model is usable. In Figure 2, it can also be seen how the loss and accuracy changes over the course of training

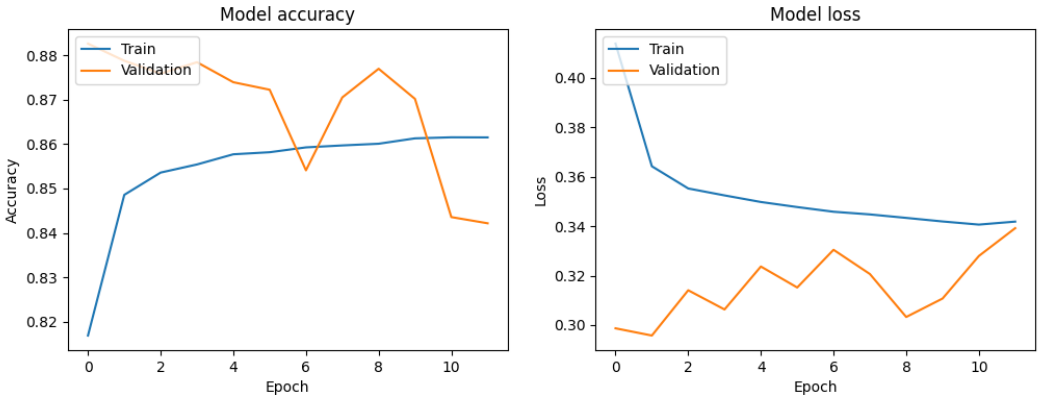

Figure 2: Generalized Model Training Metrics

Now attention can be given to the model with tuned hyperparameters. It was constructed in the same way as the baseline model but allowed many more iterations to train diverse sets of hyperparameters to identify the optimal combination. Without looking at any numbers, it would be expected that this model would have a greater model accuracy than the baseline algorithm. Analyzing the confusion matrix can help determine whether this is the case:

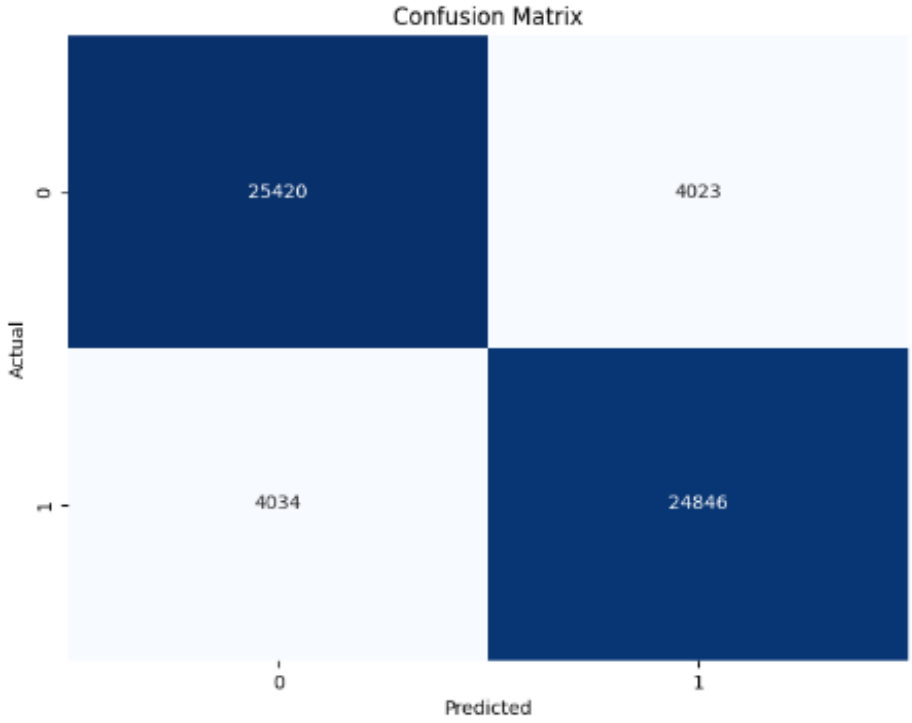

Figure 3: Tuned Model Confusion Matrix

From this confusion matrix, it can be found that the accuracy is 0.8619. This means that the model correctly predicts the class of a ship 86.19% of the time. This accuracy is nearly 2% greater than the non-tuned model! While this is not an astronomical feat, this improvement can be incredibly useful in applications where misclassifying can lead to catastrophic consequences, such as identifying cancerous cells or autonomous vehicles misclassifying road signs. In Figure 4, it can also be seen how the loss and accuracy changes over the course of training:

Figure 4: Tuned Model Training Metrics

Additional metrics that can be used to analyze the model's usefulness are precision and recall. Precision is defined as how accurately the model predicts events, so this value should be high. Recall tells an analyst how accurately the model classifies actual events, so this value should be close to 1 as well. First looking at the precision for fishing ships, it is calculated by using: (24,846) / (24,846 + 4,023) = 0.8606. The recall is calculated using the following: (24,846) / (24,846 + 4,034) = 0.8603. These values are all greater than the metrics calculated in the generalized model, which means that the tuned model is more useful when measured across all the desired metrics. Although only 6 hyperparameters were tested across 2 levels, this still provided 64 models for testing taking over 10 hours to complete. The results yield that the best

set of hyperparameters is as follows; Optimizer: Adam, Dropout Rate: 0.5, Neurons in Layer 1: 64, Neurons in Layer 2: 32, Neurons in Layer 3: 32, and Neurons in Layer 4: 16.

Conclusion

Using hyperparameter tuning, a generalized model experienced improvement in its metrics across the board. This demonstrates the power of hyperparameter tuning, and how a simple setup can result in a more useful model. The obvious downside to such a method is that the more hyperparameters one chooses to tune in their model, the longer the training of that model will take, as it needs to test not only every hyperparameter, but every combination of hyperparameters. Therefore, it is recommended that light hyperparameter tuning using only between 2 and 4 hyperparameters be done for models that seek slight improvement gain without using large amounts of time. For models that have more serious consequences, such as applications that can directly affect the health and safety of humans, more extensive hyperparameter tuning is highly recommended to ensure the best iteration of a model can be produced.

Relation to Thesis

Breaking away from standard essay format, my thesis will have a large focus on hyperparameter tuning. While the methods used in my thesis will be more rigorous than selecting a few hyperparameters to try to work with, this was still a great introduction as I learned that some hyperparameters (like dropout rate) can be broken up into decimals to use in a grid search fashion, while some (like the optimizer types) must be specifically and intentionally chosen to be tested.

Bibliography

Global Fishing Watch | Data download portal. “Global Fishing Watch | Data Download Portal.” Accessed June 11, 2024. https://globalfishingwatch.org/.