Fish Biomass Prediction

STA 561 Final Project, Spring 2023

Eli Gnesin (UID: 1172961)
Alonso Guerrero Castaneda (UID: 1194613)
Thomas Misikoff(UID: 1166813)
Sanskriti Purohit (UID: 1179957)
Will Tirone (UID: 1130904)

Duke University
April 26th, 2023

1 FAQ

1.1 What are the current/traditional methods of measuring Fish Biomass?

Fish biomass assessment, or Fishery Stock assessment, is currently conducted using a wide array of different statistical methods and models depending on the available information. When minimal data exists about a fish species' stock, methods such as Depletion Corrected Average Catch [3] are used, which is simply the proportion of the sum of catches to the number of years and an added "windfall ratio" to account for a potentially unsustainable fall in species stock. Such methods are nominally useful for indicating the relative shift of fish harvest year-to-year, and therefore can provide a recommended maximum harvest. They cannot, however, be used to determine whether or not a stock is considered "overfished" and thus are not strong methods for guaranteeing sustainable fish stock [7].

In cases when more data is available for a fish species, the range of available modeling options increases to include methods such as Aggregate Biomass Dynamics models, which evaluate the size and relative growth of a stock over time, and index based assessment models which use indices constructed from data collected through surveys and records in order to explore trends over time, and therefore determine sustainable catch levels. These methods, though they are more data-driven, also have their own flaws and concerns. With Aggregate Biomass Dynamics models, a major concern is that these models are most robust when there are periods of relative high and low abundance and catch to provide estimates, but not every fish stock will go through these cyclic periods. Index-based methods, in contrast, are not dependent on any structure of the data, but are largely heuristically driven, with decisions made based on whether index values match some critical threshold decided by expert, and further, like the data-poor methods above, cannot generally evaluate whether a stock is considered "overfished" [7].

1.2 What does it mean to be "overfished"? How does illegal fishing factor into this?

Overfishing is, as the name suggests, the process of catching more fish than can sustainably be replenished by the remaining breeding population[6]. Due to overfishing, fish populations are unable to recover back to initial levels during non-fishing seasons, leading to depleted stock in future fishing seasons. Because of this, it is in the best interest of governments and organizations which manage fishing rights to protect against overfishing through regulation. As noted above, however, most current methods are unable to evaluate concretely whether or not fish stock is being overfished, and therefore are not useful in assessing the sustainability of current fishing measures. Furthermore, due to illegal or unreported fishing, it is possible that current methods may overstate actual fish stock, since they may not successfully or completely account for the confounding factor of fish stock being removed from the overall supply due to illegal activities [1].

Given the heavy negative consequences associated with overfishing, and adjacently by illegal fishing activities, governments often seek to regulate their fishing industries, at least for specific species, to try to ensure sustainable populations [1]. With our method, which relies on oceanographic data, we hope to supplement current methods to give governments a better overall understanding of the fish population in the area, even if they are unable to gather complete assessments of fish stock through catch surveys. Although our methods may also suffer from similar concerns of high illegal or unreported fishing activity as a confounding factor, it is nonetheless useful to offer an adjacent method that could either support catch reports by fishermen or cast doubt as to the accuracy of their predictions.

1.3 How was the data gathered and how reliable is it? What is the legitimacy of the publishing organization?

We have two sources of data, historical catch data from the National Oceanic and Atmospheric Administration (NOAA) from the Alaska Region, and ocean temperature and salinity measurements from the Hybrid Coordinate Ocean Model (HyCOM) which "is a multi-institutional effort sponsored by the National Ocean Partnership Program (NOPP), as part of the U. S. Global Ocean Data Assimilation Experiment (GODAE), to develop and evaluate a data-assimilative hybrid isopycnal-sigma-pressure (generalized) coordinate ocean model." We believe both of these are the most accurate freely-available data sources given the complex nature of measuring ocean properties. We downloaded a .csv of historical catch data from the NOAA, but the HyCOM data was not as readily available. To gather it, we performed a web scraping (detailed in Listing 1 in the Technical Appendix). The only notable issue was that the longitude of the 2019 data had an incorrect sign which resulted in a point in the southern hemisphere rather than northern. We verified manually that this was an error by comparing to previous years, and viewing the

true coordinates on a globe. When we confirmed this was an error, we multiplied the longitude by -1 and verified previous years to ensure it was only happening in 2019.

1.4 How can this process be applied to other fish or oceanic species? What data is necessary in order to produce the model?

This is readily applicable to other species of fish, since we used fish-agnostic features. We would encourage input from subject matter experts like marine biologists and fishermen to determine the most useful data to use and predict. The most significant deviation we envision is based on fish species habitat, since different species have significantly different living environments that would effect whether or not our choice of features accurately predict biomass. We believe we would need different measurements if we were predicting something like crab biomass in a given area since they live at the bottom of the ocean compared to fish that live at different ocean depths. For different species, we would also want to take into account the expected density of fish biomass since that may require a wider measurement area and would be less dense in the open ocean compared to fish that live close to the coast or near the surface of the ocean.

1.5 Why use Gradient Boosting? Why not use a simpler model?

We first considered the sorts of current and traditional methods for assessing fish biomass as explained in Section 1.1, and noted the general successes and flaws of each model type. First, we recognized that, since our goal was to build a model that can estimate fish biomass from spatial-temporal features, we ruled out linear modeling techniques, since we did not believe there to be a linear relationship between features such as latitude and longitude and salinity and fish biomass. Similarly, we wanted to build a relatively data-limited model, compared to the more extensive methods of prediction such as the Aggregate Biomass Dynamics models explained in Section 1.1. Finally, we wanted to consider a machine learning model, in the hope that we could build a model which could bring together index-based data from catch logs, together with oceanographic data, to build a model that could supplement, if not surpass, current methods.

With this considered, we moved to consider machine learning models which were regression oriented but non-linear and could be predictive even with low-dimensional data without quickly being overfit. Gradient boosting, through its use of both subsampling and shrinkage-based regularization, is relatively resistant to overfitting, which is a necessity in ensuring that our model could be applied to the walleye pollock over multiple years, and in ensuring that our modeling techniques could be generalized to other fish species [4].

1.6 How can we be sure that the model is giving reasonable results?

Through the SHAP values obtained for the model, we can observe the individual impacts of the features utilized to make the predictions. The variable "depth" is discovered to have the most significant impact on our target variable, followed by "longitude" and "latitude". Considering the fact that we only utilized 5 features in this model, we obtained a 52% value for R-squared in our training set, and a close 41% in our test set. This implies that almost half of the variability in the biomass of the pollock can be explained by just these 5 variables. Obtaining and transforming species-specific or other geographic features could assist in a more effective prediction.

1.7 What are some potential pitfalls in this process?

One of the major concerns about our model is that we are training our model using the estimates from individual catches on the species (here, the Walleye Pollock). In doing so, the accuracy of our predictions relies, at least in part, on the assumption that the estimates of fish biomass (or fish size) provided by individual fishermen are correct.

We also believe that the more extreme (high) values of catch can be precisely predicted by collection and usage of more variables affecting the habitat of the fish. Further, we believe that domain and expert knowledge should be consulted to be able to achieve an optimum method for data pre-processing. This will also aid in minimizing the common problem of bad extrapolation in tree models.

2 Technical Appendix and Proof of Concept

With our goal in mind of predicting fish biomass, we started by gathering spatio-temporal data for Walleye Pollock in the Gulf of Alaska, which we extracted from the NOAA Fisheries website [5], and includes data like `haulID, Latitude, Longitude, Depth, Year` and `wtcpue`, which is the survey catch measured in kilograms per hectare. Data from the NOAA is available for the following years: 1996, 1999, 2003, 2005, 2007, 2009, 2011, 2013, 2015, 2017 and 2019. Then, to add temperature and salinity covariates to predict biomass, we web-scrape these from HyCOM (Hybrid Coordinate Ocean Model) using their NCSS request URL. Since there is a large amount of data, we decided to get a sample from the first day of each summer month. Including more data would improve our model, though we don't have a time stamp associated with the data from the NOAA. We used the following geographic region: North: 60.32, South: 52.41, West: -170, East: -132.5. The following code performed the web-scraping:

```
# Define the list of years and months
year_list = [1996, 1999, 2003, 2005, 2007, 2009, 2011, 2013, 2015, 2017, 2019]
month_list = [7, 8, 9]
this_folder = os.getcwd()
data_folder = this_folder + '\\temp' # '/temp' on a Unix machine

if not os.path.exists('ocean_data.csv'):
  # create data folder if it doesn't exists
  if not os.path.exists(data_folder):
    os.mkdir(data_folder)

  # Create an empty DataFrame to store the results
  df_all = pd.DataFrame()

  # Loop over each desired year and month, and add a progress bar
  for year in year_list:
      for month in tqdm(month_list, desc=f'{year}'):
          # Define the date string
          date_str = f'{year}-{month:02d}-01T09:00:00Z'
          # Define the URL with the fixed parameters
          if year <=2015:
              url = "https://ncss.hycom.org/thredds/ncss/GLBv0.08/expt_53.X/data/"+str(year)+"?var=
    salinity_bottom&var=water_temp_bottom&north=60.32&west=-170&east=-132.5&south=52.41&horizStride
    =1&vertCoord=&accept=netcdf4"
          elif year == 2017:
              url = "https://ncss.hycom.org/thredds/ncss/GLBv0.08/expt_57.7?var=salinity_bottom&var
    =water_temp_bottom&north=60.32&west=-170&east=-132.5&south=52.41&horizStride=1&vertCoord=&
    accept=netcdf4"
          else:
              url = "https://ncss.hycom.org/thredds/ncss/GLBv0.08/expt_93.0/ts3z?var=
    salinity_bottom&var=water_temp_bottom&north=60.32&west=-170&east=-132.5&south=52.41&horizStride
    =1&vertCoord=&accept=netcdf4"

          my_file = os.path.join(this_folder, 'temp', f'example-{date_str[:10]}.nc4')
          # Add the date parameter to the URL
          url_day = f'{url}&time={date_str}'
          # Download the data and save it to a file
          downloaded_obj = requests.get(url_day)
          with open(my_file, "wb") as file:
              file.write(downloaded_obj.content)
              del downloaded_obj
          # Open the NetCDF file and convert it to a pandas DataFrame
          ds = xr.open_dataset(my_file)
          df_temp = ds.to_dataframe().reset_index()
          # Append the DataFrame to the overall DataFrame
          df_all = pd.concat([df_all, df_temp], ignore_index=True)

  df_all.to_csv('ocean_data.csv', index=False)
```

Listing 1: Web Scraping Ocean Data

Now, our goal was to merge the two dataframes, so we had to clean the data to ensure it would join smoothly. Problematically, the 2019 data from HyCOM had an incorrect sign on the longitude of the data, which results in a coordinate that is on the opposite side of the globe from where we want. After verifying that this was not happening

for other datasets and that the latitude was correct, we multiplied through by -1 to correct this. Joining the datasets based on precise location proved impossible since latitude and longitude were measured down to 4 or 5 digits so there weren't enough matches in both datasets. Our first attempt was to take the lat / lon from `wp_gulf_alaska` (our dataframe of fish biomass) and find the closest lat / lon match in `ocean_data_df` (the salinity and temperature measurements from the ocean). However, for a given year of ocean data, there were hundreds of thousands of observations, so computing the distance to each point would take far too long. To make this achievable, we found the minimum and maximum longitudes needed for the Gulf of Alaska, and subset `ocean_data_df` by those values. Then, we computed the Haversine distance for the points (though noted scholar Mark Sargent may choose Euclidean distance), chose the closest, and merged the dataframes.

```
def dist(lat1, lon1, lat2, lon2):
    """
    https://medium.com/analytics-vidhya/finding-nearest-pair-of-latitude-and-longitude-match-using-python-ce50d62af546
    """
    # convert decimal degrees to radians
    lat1, lon1, lat2, lon2 = map(radians, [lat1, lon1, lat2, lon2])
    # haversine formula
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
    c = 2 * asin(sqrt(a))
    # Radius of earth in kilometers is 6371
    km = 6371 * c
    return km

def find_nearest(lat, lon, year):
  year_ocean_data_df = grouped_ocean_data[grouped_ocean_data['year'] == year]


  distances = year_ocean_data_df.apply(
      lambda row: dist(lat, lon, row['lat'], row['lon']),
      axis=1)
  return year_ocean_data_df.loc[distances.idxmin(), 'ocean_data_id']

sub_wp_gulf_alaska['ocean_data_id'] = sub_wp_gulf_alaska.apply(
    lambda row: find_nearest(row['lat'], row['lon'], row['year']),
    axis=1)

merge_df = pd.merge(sub_wp_gulf_alaska, grouped_ocean_data, on='ocean_data_id', how='left',
    suffixes=('_wp', '_ocean'))

merge_df.head()
```

Listing 2: Distance Calculations and Merge

	haul_id	stratum	lat_wp	lon_wp	depth	year_wp	wtcpue	ocean_data_id	lat_ocean	lon_ocean	year_ocean	salinity_bottom	water_temp_bottom
0	023-199601-003	210	52.59615	-169.4421	235	1996	9.0793	478	53.0	-169.0	1996	33.791651	3.926612
1	023-199601-004	111	52.78445	-168.7133	109	1996	0.3296	478	53.0	-169.0	1996	33.791651	3.926612
2	023-199601-005	10	52.85188	-168.6160	95	1996	646.7679	478	53.0	-169.0	1996	33.791651	3.926612
3	023-199601-006	10	52.98299	-168.2722	106	1996	256.1357	488	53.0	-168.0	1996	33.666818	3.898750
4	023-199601-007	210	52.96844	-167.5347	227	1996	3.0989	488	53.0	-168.0	1996	33.666818	3.898750

Figure 1: Sample Data From the Combined DataFrame

To fit the model, we set aside the 2019 data as a test set and used the other years for training. Further, we transformed our target variable by converting it into logarithmic values to produce a more normally skewed

distribution. Then, we performed hyperparameter optimization to produce a model that achieved the minimum RMSE. [2] The code to accomplish this is below:

```
y = merge_df[['year_wp', 'wtcpue']]
X = merge_df[['year_wp','lat_wp', 'lon_wp', 'depth','salinity_bottom','water_temp_bottom']]

## Log transformation to make the target variable follow normal distribution.
y_train = np.log(y[y['year_wp'] != 2019]['wtcpue']+1)
y_test = np.log(y[y['year_wp'] == 2019]['wtcpue']+1)

X_train = X[X['year_wp'] != 2019].drop('year_wp', axis=1)
X_test = X[X['year_wp'] == 2019].drop('year_wp', axis=1)

space = {'max_depth': scope.int(hp.quniform("max_depth", 1, 10, 1)),
        'n_estimators': hp.choice('n_estimators', np.arange(100, 1000, 100, dtype="int")),
        'learning_rate': hp.uniform('learning_rate', 0, 1),
        'gamma': hp.uniform ('gamma', 0,1),
        'reg_alpha' : hp.uniform('reg_alpha', 0,50),
        'reg_lambda' : hp.uniform('reg_lambda', 10,100),
        'colsample_bytree' : hp.uniform('colsample_bytree', 0,1),
        'min_child_weight' : hp.uniform('min_child_weight', 0, 5)}

def hyperparameter_tuning(space,X_train=X_train,y_train=y_train):
    model = xgb.XGBRegressor(**space)

    #Define evaluation datasets.
    X_train, X_valid, y_train, y_valid = train_test_split(X_train, y_train, random_state=3,
    train_size=.8)
    evaluation = [(X_train, y_train), (X_valid, y_valid)]

    #Fit the model. Define evaluation sets, early_stopping_rounds, and eval_metric.
    model.fit(X_train, y_train,
            eval_set=evaluation, eval_metric="rmse",
            early_stopping_rounds=100,verbose=0)

    #Obtain prediction and rmse score.
    pred = model.predict(X_valid)
    rmse = mean_squared_error(y_valid, pred, squared=False)
    print ("SCORE:", rmse)

    #Specify what the loss is for each model.
    return {'loss':rmse, 'status': STATUS_OK, 'model': model}

trials = Trials()
best = fmin(fn=hyperparameter_tuning,
            space=space,
            algo=tpe.suggest,
            max_evals=30,
            trials=trials)

regressor = trials.results[np.argmin([r['loss'] for r in trials.results])]['model']

regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
```

Listing 3: Fitting the Gradient Boosting Model

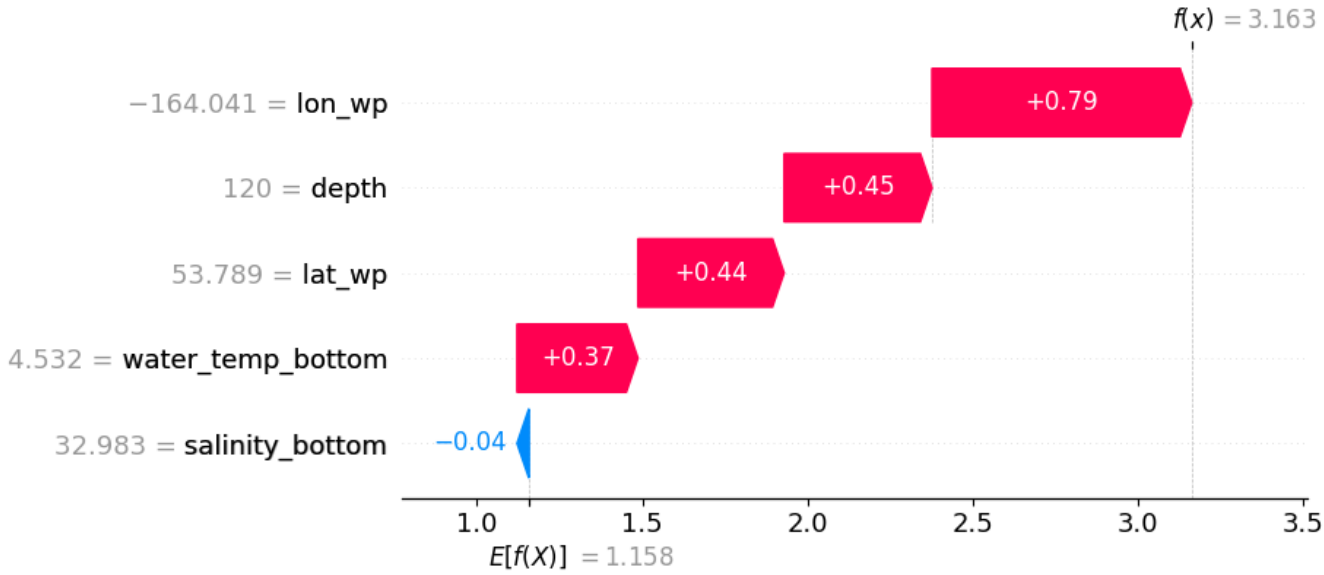
This resulted in training MSE of 0.74 and test MSE of 1.65. Further, we obtained the R-squared value for our training set as 0.52 and test set as 0.41. However, since Gradient Boosting is not interpretable, we turn to SHAP Values to offer a perspective on how well our model fits.

In the waterfall plot below, for this particular observation with an actual value of 1.158, it can be observed that having a very low (negative) value in longitude increases the value for our prediction, implying a negative correlation between longitude and the mass of the observed fish. Further, positive values in depth, latitude and water temperature also seem to increase the value obtained by our function. Finally, a high value in salinity, seems to have a negligibly negative impact on the prediction.

```
X = X.drop('year_wp', axis=1)
```

```
X100 = shap.utils.sample(X, 100)
explainer_boost = shap.Explainer(regressor, X100)
shap_values_boost = explainer_boost(X)
shap.plots.waterfall(shap_values_boost[sample_ind])
```

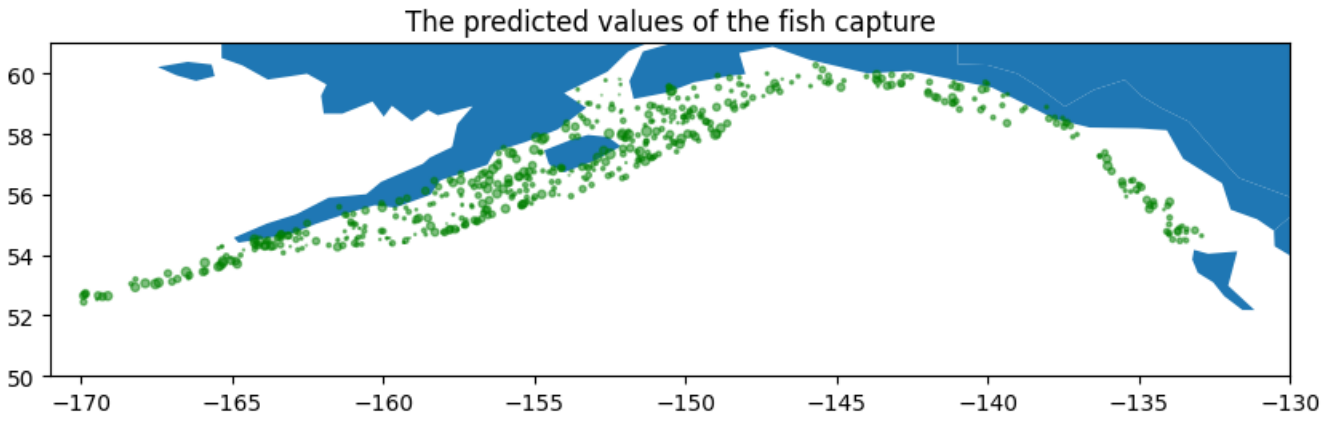
Listing 4: Calculating and plotting SHAP values.

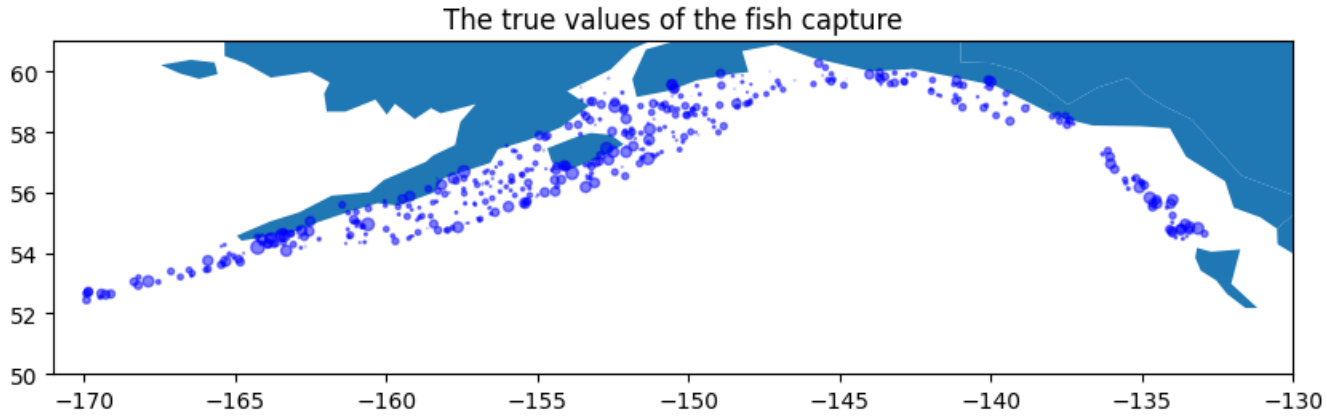



```
geometry = [Point(xy) for xy in zip(X_test['lon_wp'], X_test['lat_wp'])]
gdf = GeoDataFrame(X_test, geometry=geometry)

#this is a simple map that goes with geopandas
world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))
gdf.plot(ax=world.plot(figsize=(10, 6)), markersize =abs(y_pred)*5, color='green',alpha=0.5)
plt.xlim([-171, -130])
plt.ylim([50, 61])
plt.title("The predicted values of the fish capture")
gdf.plot(ax=world.plot(figsize=(10, 6)), markersize=abs(y_test)*5, color='blue',alpha=0.5)
plt.xlim([-171, -130])
plt.ylim([50, 61])
plt.title("The true values of the fish capture")
plt.show()
```

Listing 5: Comparing the predicted outcomes.





References

[1] *Global Consequences of Overfishing*. Feb. 2009. URL: https://www.dfo-mpo.gc.ca/international/isu-global-eng.htm.

[2] *Hyperopt and XGBRegressor - Bayesian Optimization*. 2021. URL: https://www.kaggle.com/questions-and-answers/206121.

[3] Alec D. McCall. *Depletion-corrected average catch: a simple formula for estimating sustainable yields in data-poor situations*. URL: https://academic.oup.com/icesjms/article/66/10/2267/682739 (visited on 04/15/2023).

[4] Alexey Natekin and Alois Knoll. "Gradient Boosting Machines, A Tutorial". In: *Frontiers in neurorobotics* 7 (Dec. 2013), p. 21. DOI: 10.3389/fnbot.2013.00021.

[5] NOAA. *NOAA Fisheries Data*. URL: https://apps-st.fisheries.noaa.gov/dismap/DisMAP.html#single-species-distributions (visited on 04/15/2023).

[6] *Overfishing: The most serious threat to our oceans*. URL: https://www.edf.org/oceans/overfishing-most-serious-threat-our-oceans.

[7] *Stock assessment model descriptions*. URL: https://www.fisheries.noaa.gov/insight/stock-assessment-model-descriptions.