HW7+8+9

April 12, 2022

0.1 HW 7: Data Description & Preprocessing with Input Data Visualization

0.1.1 OCEN 460

0.1.2 Team: _/Sample_Text/

0.1.3 Members: Nate Baker and James Frizzell

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import pathlib
import os

%matplotlib inline
#Github: https://github.com/jafrizzell/coral-prediction.git
#Describe Datasets and project idea
```

- The World Ocean Atlas (WOA) data cannot be shown in it's raw for here because it is too large to be shared in teh github repository where the data is stored. The metadata for it looks as follows. Replace temperature with "salinity" or "dissolved oxygen" for the other two datasets collected from WAO.

Latitude | Longitude | Temperature@0m depth (Celsius) | Temp@5m | Temp@10m | Temp@15m |...| Temp@5500m

- The Deep Sea Coral Data (DSC) has the following metadata

Latitude | Longitude | Depth (m) |

0.1.4 1. The deep sea coral dataset reports latitude and longitude of known coral growth locations with the depth at which the coral is growing. The World Ocean Atlas reports depth measurements in increments of 5 meters for depths of 0 to 100 meters, 10 meters for 100 to 500 meters, 50 meters for 500 to 2000 meters, and in 100 meters for greater than 2000 meters. The following code was used and adjusted to round the Deep Sea Coral dataset to match this convention.

[ ]:
```
path = 'C:/Users/jafri/Documents/GitHub/coral-prediction/processed_data/
 ↪deep_sea_corals_rounded.csv'

raw = pd.read_csv(path)
```

```
def round_depth(x, base):
    return int(base * round(float(x)/base))


raw['depth'] = raw['depth'].apply(lambda x: round_depth(x, base=5))

raw = raw[raw.depth >= 0]
print(len(raw))
raw.to_csv('C:/Users/jafri/Documents/GitHub/coral-prediction/processed_data/
 ↪deep_sea_corals_rounded_depthcorr.csv')
```

0.1.5 2. The following code aligns latitude and longitude values from the Deep Sea Coral dataset with the lat/long values from the WOA dataset with a tolerance of 0.5 degrees. Second_param file can be changed to indicate the oceanographic variable of interest. WOA data is right-joined to DSC data for further preprocessing.

0.1.6 The code yields a .csv file that contains the DSC data and the WOA data. The WOA data is depth-stratified.

[ ]:
```
import geopandas

coral = 'D:/TAMU Work/TAMU 2022 SPRING/OCEN 460/depthtempsal_short2.csv'
second_param = 'D:/TAMU Work/TAMU 2022 SPRING/OCEN 460/woa18_all_O00mn01.csv' ␣
 ↪# "O00mn01" indicates O2 data

raw_coral = pd.read_csv(coral)
raw_coral = geopandas.GeoDataFrame(raw_coral, geometry=geopandas.
 ↪points_from_xy(raw_coral.longitude, raw_coral.latitude))
raw_coral.depth = raw_coral.depth.astype(float)
raw_coral.latitude = raw_coral.latitude.astype(float)
raw_coral.longitude = raw_coral.longitude.astype(float)

raw_param = pd.read_csv(second_param)
raw_param = raw_param.astype(float)
raw_param = geopandas.GeoDataFrame(raw_param, geometry=geopandas.
 ↪points_from_xy(raw_param.longitude, raw_param.latitude))


depth_sal = raw_coral.sjoin_nearest(raw_param, max_distance=0.5)

depth_sal.to_csv('D:/TAMU Work/TAMU 2022 SPRING/OCEN 460/depthtempsaloxy.csv',␣
 ↪index=False)
```

0.1.7 3. To resolve the stratified nature of the WOA data, the following code is used to select the corresponding WOA column for the DSC depth of interest.

[ ]:
```
path = 'D:/TAMU Work/TAMU 2022 SPRING/OCEN 460/depthtempsaloxy.csv'

raw = pd.read_csv(path)
raw = raw[raw['depth'] <= 5500]
skipped = 0
for i in range(len(raw)):
    try:
        depth = str(raw['depth'][i])
        raw['oxygen'][i] = raw[depth][i]
    except KeyError:
        skipped+=1
        pass

print("skipped:", skipped)

raw.to_csv('D:/TAMU Work/TAMU 2022 SPRING/OCEN 460/depthtempsaloxy_short.csv',␣
 ↪index=False)
```

0.1.8 4. The following code determines the maximum depth for each lat/long pair in the WOA dataset. These datapoints were then used to create a control dataset describing where coral is not present, in order to compare to the DSC dataset. Code displayed in sections 2 and 3 were used to add the temperature, salinity, and oxygen variables to the control dataset.

[ ]:
```
path = 'D:/TAMU Work/TAMU 2022 SPRING/OCEN 460/woa18_decav_t00mn04.csv'

raw = pd.read_csv(path)
depth = []

for i in range(len(raw)):
    for j in range(103):
        curr = raw.iloc[i, -1-j]
        plus = raw.iloc[i, -2-j
        if j == 0 and np.isfinite(curr):
            depth.append(raw.columns[-1])
            break
        elif np.isnan(curr) and np.isfinite(plus):
            depth.append(raw.columns[-2-j])
            break
        elif j == 102:
            depth.append(raw.columns[-1])
print(len(depth))
print(len(raw.latitude))
out = pd.DataFrame({'latitude': raw.latitude,
```

```
                    'longitude': raw.longitude,
                    'depth': depth})

out.to_csv('D:/TAMU Work/TAMU 2022 SPRING/OCEN 460/depths.csv', index=False)
```

Ultimately, the final dataset had the following metadata

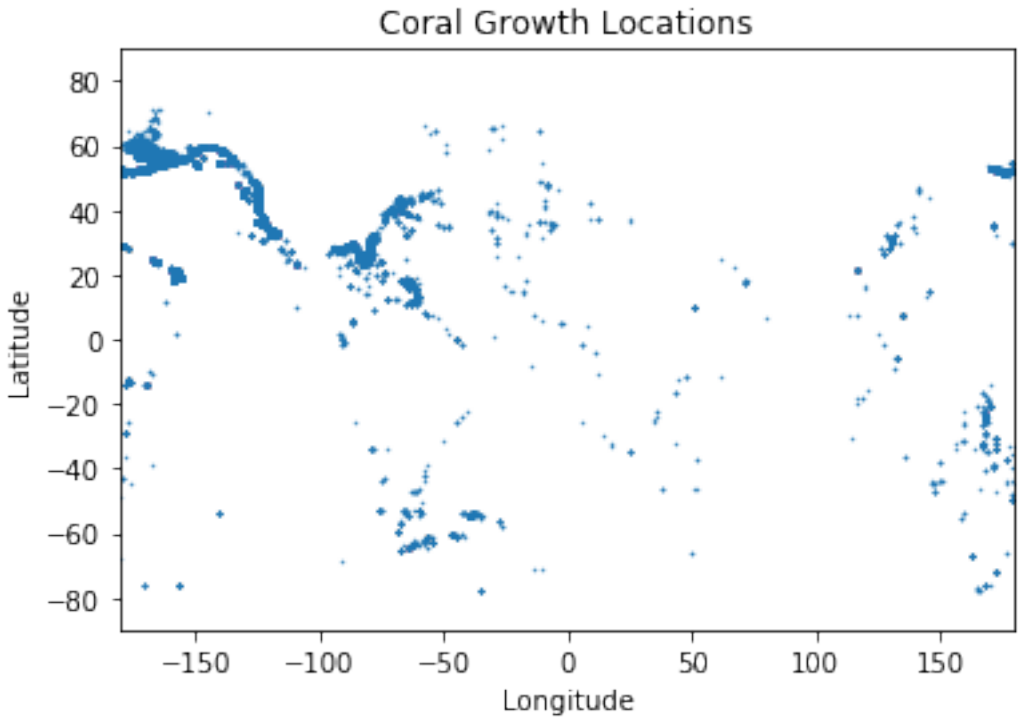
Latitude | Longitude | Depth (m) | Temperature (c) | Salinity (ppt) | Dissolved Oxygen (umol/kg)

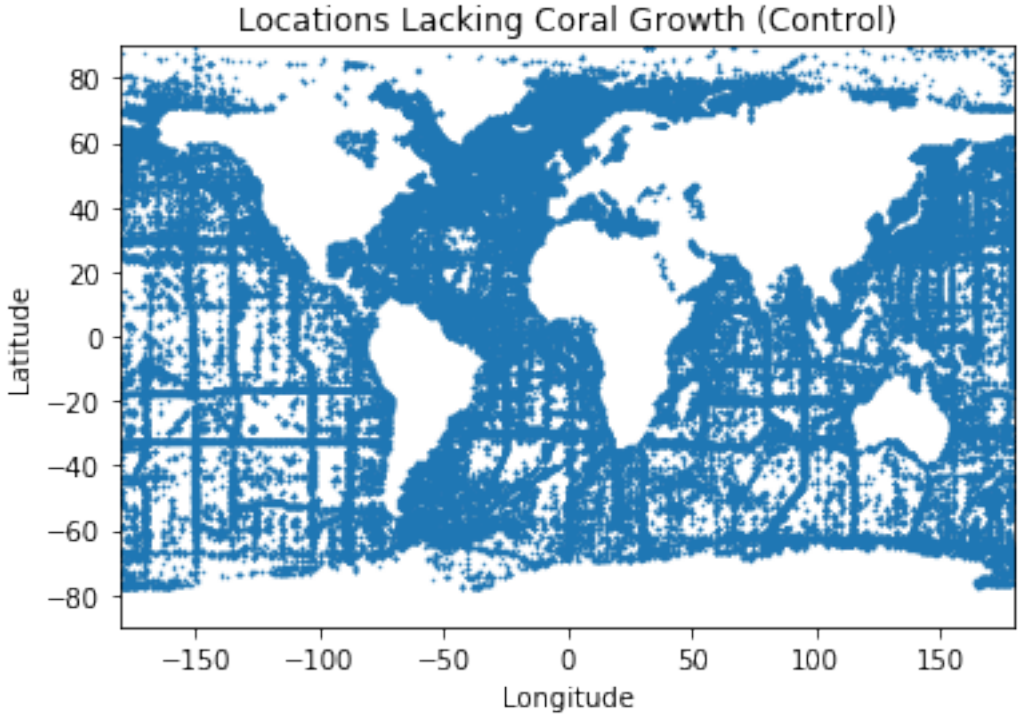
0.1.9 5. The following code visualizes the two datasets.

[2]:
```
#Reprocessed Data for Visualization
path = str(pathlib.Path(os.getcwd())) +␣
 ↪'\processed_data\combined_data_truncated.csv'
raw = pd.read_csv(path)
print(raw.describe())

#Visualization
coral_present_bool = raw[raw.coral_present == 1]
plt.scatter(coral_present_bool['longitude'], coral_present_bool["latitude"], s=␣
 ↪0.2)
plt.title("Coral Growth Locations")
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.xlim([-180,180])
plt.ylim([-90,90])
plt.show()

coral_missing_bool = raw[raw.coral_present == 0]
plt.scatter(coral_missing_bool['longitude'], coral_missing_bool["latitude"], s=␣
 ↪0.2)
plt.title("Locations Lacking Coral Growth (Control)")
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.xlim([-180,180])
plt.ylim([-90,90])
plt.show()

print("Number of Coral Growth Datapoints:", len(coral_present_bool))
print("Number of Datapoints with no Coral Growth", len(coral_missing_bool))

plt.scatter(raw.longitude, raw.latitude, s=0.2, c=raw.depth)
plt.title("Cumulative Dataset, Colored By Depth")
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.xlim([-180,180])
plt.ylim([-90,90])
plt.colorbar()
```

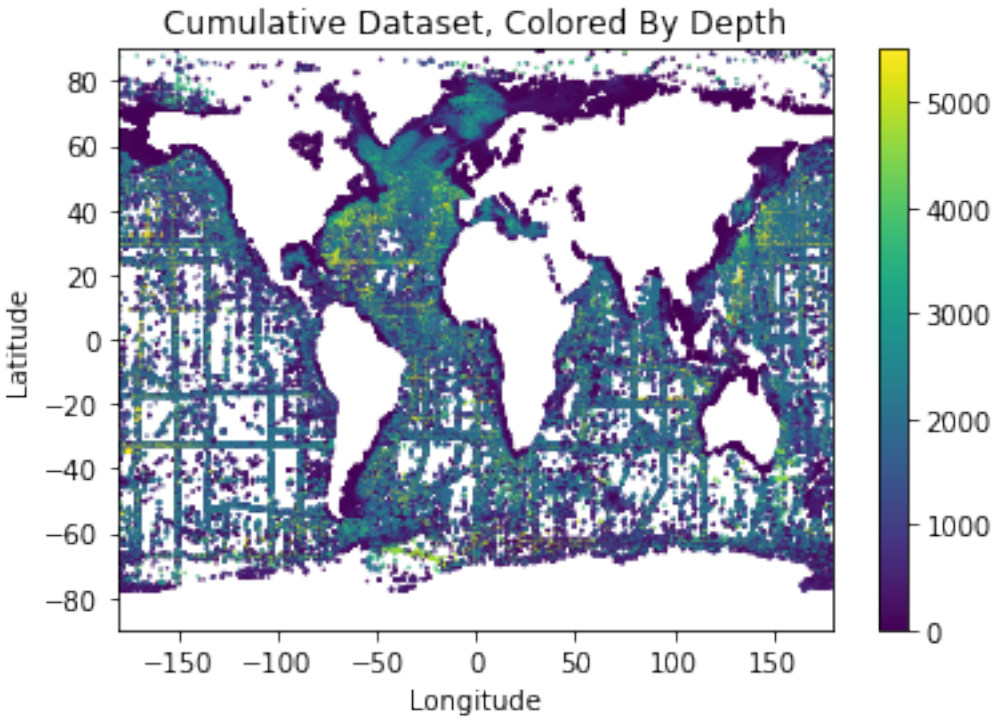
```
plt.show()
plt.scatter(raw.longitude, raw.latitude, s=0.2, c=raw.temperature)
plt.title("Cumulative Dataset, Colored By Temperature")
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.xlim([-180,180])
plt.ylim([-90,90])
plt.colorbar()
plt.show()
plt.scatter(raw.longitude, raw.latitude, s=0.2, c=raw.salinity)
plt.title("Cumulative Dataset, Colored By Salinity")
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.xlim([-180,180])
plt.ylim([-90,90])
plt.colorbar()
plt.show()
plt.scatter(raw.longitude, raw.latitude, s=0.2, c=raw.oxygen)
plt.title("Cumulative Dataset, Colored By Oxygen")
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.xlim([-180,180])
plt.ylim([-90,90])
plt.colorbar()
plt.show()
```

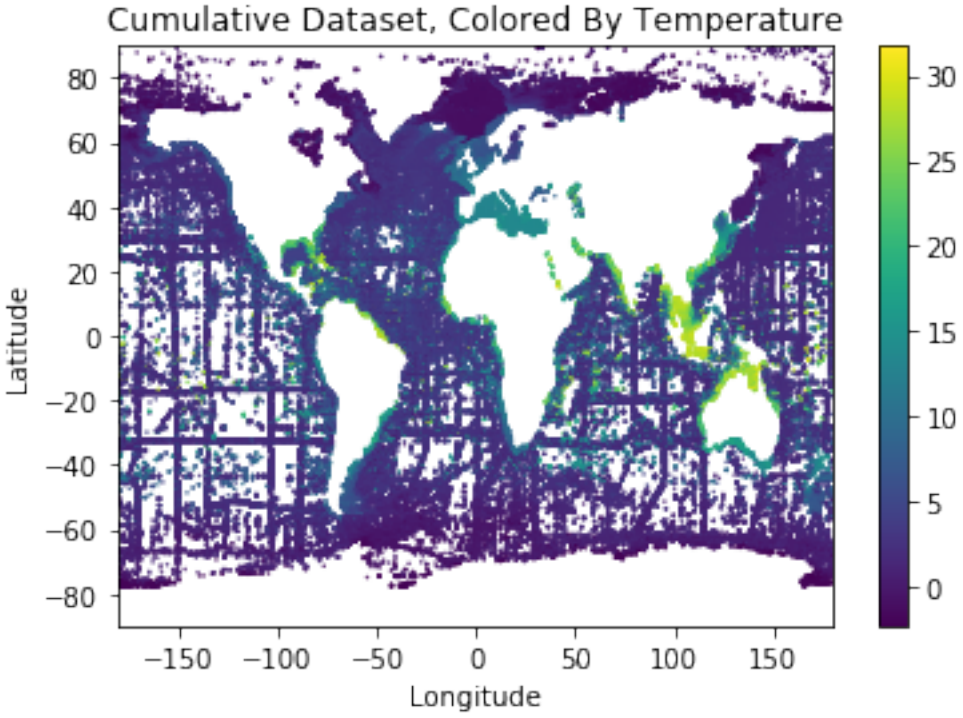
```
       coral_present       latitude      longitude          depth  \
count  341773.000000  341773.000000  341773.000000  341773.000000
mean        0.582340      23.237462     -68.404081    1066.013070
std         0.493174      33.783962      96.027313     989.540361
min         0.000000     -77.875000    -179.989750       0.000000
25%         0.000000      16.625000    -124.339620     235.000000
50%         1.000000      35.641580    -119.498760     850.000000
75%         1.000000      40.811190     -25.625000    1750.000000
max         1.000000      89.875000     179.989750    5500.000000

             round_d    temperature       salinity         oxygen
count  341773.000000  341773.000000  341773.000000  341773.000000
mean     1067.012915       5.191193      34.392171      42.294298
std       989.771161       4.584556       1.343642      29.076344
min         0.000000      -2.271000       0.000000       0.199000
25%       225.000000       2.452000      34.228000      12.890000
50%       850.000000       3.878000      34.520000      41.558000
75%      1750.000000       7.371000      34.699000      65.495000
max      5500.000000      31.751000      41.310000     132.182000
```

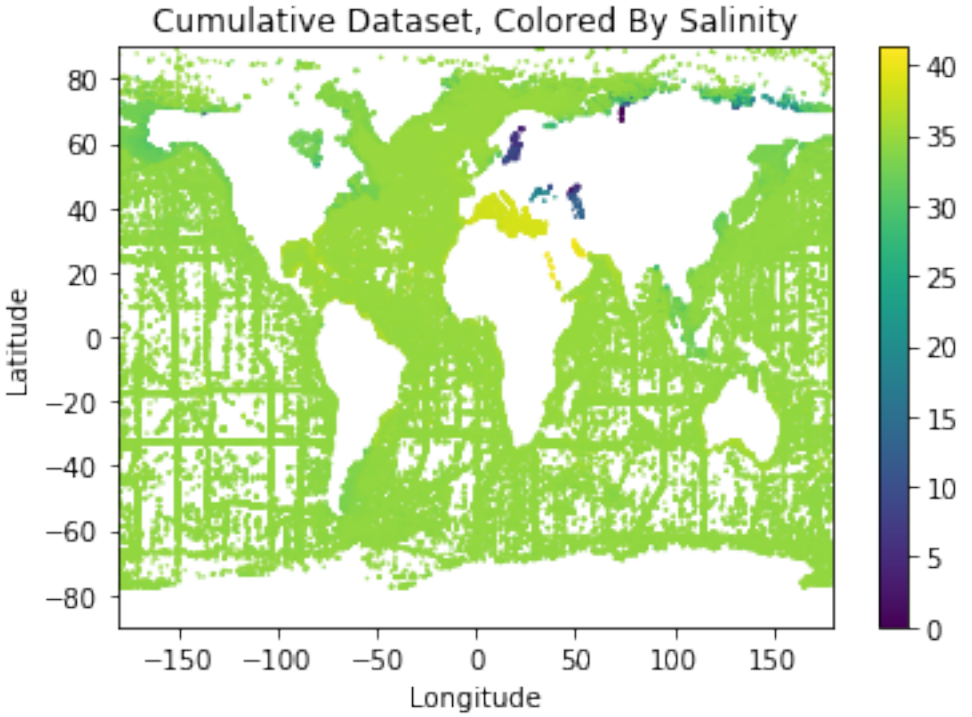

Coral Growth Locations
Latitude
Longitude
80
60
40
20
0
−20
−40
−60
−80
−150
−100
−50
0
50
100
150

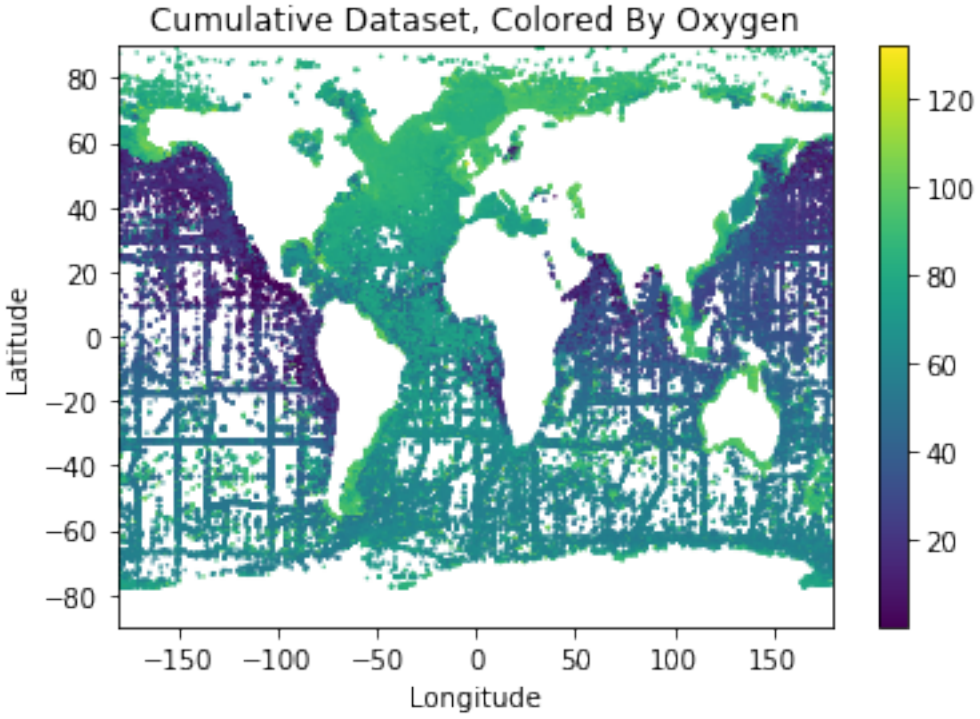

Locations Lacking Coral Growth (Control)
Latitude
Longitude
80
60
40
20
0
−20
−40
−60
−80
−150
−100
−50
0
50
100
150

```
Number of Coral Growth Datapoints: 199028
Number of Datapoints with no Coral Growth 142745
```






Cumulative Dataset, Colored By Salinity
Latitude
Longitude
80
60
40
20
0
−20
−40
−60
−80
−150
−100
−50
0
50
100
150
40
35
30
25
20
15
10
5
0


Cumulative Dataset, Colored By Oxygen
Latitude
Longitude
80
60
40
20
0
−20
−40
−60
−80
−150
−100
−50
0
50
100
150
120
100
80
60
40
20

0.2 HW 8: Pattern Extraction & Why Algorithm Chosen (+ Parameter Tuning)

0.2.1 OCEN 460

0.2.2 Team: _/Sample_Text/

0.2.3 Members: Nate Baker and James Frizzell

Overview:

The purpose of this project is to use existing data on the growth of coral to predict whether coral can grow given oceanographic conditions. The latitude, longitude, depth, temperature, salinity, and dissolved oxygen levels are used to predict a binary value with 1 meaning that coral can grow and 0 meaning that coral cannot grow.

[3]:
```
import os
import tensorflow as tf
import pandas as pd
import numpy as np
from tensorflow import keras
from tensorflow.keras import layers
import matplotlib.pyplot as plt
import RegscorePy
from math import sqrt, floor
import pathlib
from itertools import product
from scipy.stats import pearsonr
import time
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
%matplotlib inline
```

```
      ␣
↪---------------------------------------------------------------------------

       ModuleNotFoundError                       Traceback (most recent call␣
↪last)

       <ipython-input-3-4b3d92add505> in <module>()
         1 import os
   ----> 2 import tensorflow as tf
         3 import pandas as pd
         4 import numpy as np
         5 from tensorflow import keras
```

```
ModuleNotFoundError: No module named 'tensorflow'
```

Import the necessary packages. Some uncommon ones are RegscorePy, which is a custom open-source library used to calculate the Akaike Information Criterion (AIC) of tensorflow models, and itertools.product which is used to generate all combinations of elements in a list.

0.3 1) Load data in

The raw data is loaded in from a csv file and processed into a training and testing dataset.

[2]:
```
cwd = pathlib.Path(os.getcwd())
path = str(cwd.parent) + '/coral-prediction/processed_data/
 ↪combined_data_truncated.csv'

raw = pd.read_csv(path)


raw = raw.sample(frac=0.2, random_state=0)

print(raw.describe())

raw.pop('species')
raw.pop('round_d')

train = raw.sample(frac=0.8, random_state=0)
test = raw.drop(train.index)

train_features = train.copy()
test_features = test.copy()

train_labels = train_features['coral_present']
test_labels = test_features['coral_present']
train_features.pop('coral_present')
test_features.pop('coral_present')
```

```
       coral_present      latitude     longitude         depth       round_d  \
count   68355.000000  68355.000000  68355.000000  68355.000000  68355.000000
mean        0.584273     23.283085    -68.820676   1063.477800   1064.483798
std         0.492850     33.799299     95.889386    985.059453    985.321543
min         0.000000    -77.875000   -179.966080      0.000000      0.000000
25%         0.000000     16.875000   -124.591595    235.000000    225.000000
50%         1.000000     35.652870   -119.501450    855.000000    850.000000
75%         1.000000     40.811420    -26.375000   1735.000000   1750.000000
max         1.000000     89.625000    179.973800   5500.000000   5500.000000

        temperature      salinity        oxygen
count  68355.000000  68355.000000  68355.000000
mean       5.175744     34.384281     42.198043
```

```
std         4.551002      1.371827     29.077581
min        -2.248000      0.022000      0.434000
25%         2.457000     34.228000     12.890000
50%         3.878000     34.520000     41.558000
75%         7.371000     34.696000     65.169500
max        31.751000     41.310000    120.705000
```

[2]:
```
304228    0
213311    0
172619    1
141430    1
133490    1
         ..
267861    0
209754    0
192209    1
36135     1
278318    0
Name: coral_present, Length: 13671, dtype: int64
```

0.4 2) Feature evaluation - correlation coefficients

The correlation coefficients and p-values for each feature are reported. Since all p-values are <0.05, each feature selected is relevant to the model and will be kept.

It is interesting that the longitude is strongly negatively correlated to the presence of coral. This implies that at more eastern locations (say 90E - 180E) it is less likely that coral will grow. This may be a residual of other conditions, such as the much deeper waters in the eastern Pacific Ocean - since the depth is also negatively correlated.

[3]:
```
print(train.corr()['coral_present'])
print(pearsonr(train_features['latitude'], train_labels))
print(pearsonr(train_features['longitude'], train_labels))
print(pearsonr(train_features['depth'], train_labels))
print(pearsonr(train_features['temperature'], train_labels))
print(pearsonr(train_features['salinity'], train_labels))
print(pearsonr(train_features['oxygen'], train_labels))
```

```
coral_present    1.000000
latitude         0.483094
longitude       -0.623733
depth           -0.381216
temperature      0.244996
salinity        -0.075697
oxygen          -0.484223
Name: coral_present, dtype: float64
(0.48309400646187506, 0.0)
(-0.6237334337918604, 0.0)
(-0.38121618825430753, 0.0)
```

```
(0.2449964411644198, 0.0)
(-0.07569703383008823, 2.618925049563251e-70)
(-0.4842228643655442, 0.0)
```

0.5 3) Function definitions

The following functions were used during the training and evaluation of the model.

fit_and_evaluate() accepts a model architecture and fits the training data to it, and the reports the accuracy metrics based on the test dataset.

The hyperparameters selected for the training are: 20% validation split and 50 epochs training duration. These were selected by testing different configurations and choosing the hyperparameters that resulted in the most accurate model.

plot_loss() accepts the model training residuals and plots them over the training duration (number of epochs) the loss and validation loss are both shown.

add_layer() is a part of the parametric model study, which can add hidden layers to a tensorflow model by passing parameters and hyperparameters. This functionality will hopefully be bundled into a package at some point so that users can pip install the ability to do a parametric study.

build_and_compile_model() accepts the model architecture from the user and creates the tensorflow model. It then fits the model and performs the accuracy evaluations.

[4]:
```
def fit_and_evaluate(architecture):
    dnn_model = build_and_compile_model(architecture)

    history = dnn_model.fit(train_features, train_labels, validation_split=0.2,␣
↪verbose=0, epochs=50)
    plot_loss(history)

    test_results = dnn_model.evaluate(test_features, test_labels, verbose=0)
    test_predictions = dnn_model.predict(test_features).flatten()
    r2= r2_score(np.asarray(test_labels).flatten(), test_predictions)
    mae = mean_absolute_error(np.asarray(test_labels).flatten(),␣
↪test_predictions)
    aic = RegscorePy.aic.aic(np.asarray(test_labels, dtype=float).flatten(), np.
↪asarray(test_predictions).astype(float), 4+2)
    rmse = sqrt(mean_squared_error(np.asarray(test_labels).flatten(),␣
↪test_predictions))
    return dnn_model, aic, r2, mae, rmse, test_predictions
```

[5]:
```
def plot_loss(history):
    plt.plot(history.history['loss'], label='loss')
    plt.plot(history.history['val_loss'], label='val_loss')
    # plt.ylim([0, 30])
    plt.xlabel('Epoch')
    plt.ylabel('Error')
    plt.legend()
```

```
    plt.grid(True)
    plt.show()
    # pass
```

[6]:
```
def add_layer(dets, hyper, prev):
    default = ['relu']
    try:
        layer = layers.Dense(dets, activation=hyper[0])(prev)
    except IndexError:
        layer = layers.Dense(dets, activation=default[0])(prev)
    return layer
```

[7]:
```
def build_and_compile_model(arch):
    # Adjust the number of hidden layers and neurons per layer that results in␣
 ↪best fit NN
    hidden_layers = []
    inputs = keras.Input(shape=(6,))
    norm_layer = layers.BatchNormalization()(inputs)
    hidden_layers.append(inputs)
    hidden_layers.append(norm_layer)
    for i in range(num_hidden):
        if arch[i] == 0:
            pass
        else:
            layer = add_layer(arch[i], arch[num_hidden:], hidden_layers[-1])
            hidden_layers.append(layer)
            layer = layers.Dropout(rate=0.2)(hidden_layers[-1])
            hidden_layers.append(layer)
    outputs = layers.Dense(1)(hidden_layers[-1])
    hidden_layers.append(outputs)
    model = keras.Model(inputs=inputs, outputs=outputs)

    model.compile(loss='mean_absolute_error',
                  optimizer=tf.keras.optimizers.Adam(0.001))
    return model
```

0.6 4) Using the model trainer

The following code sets up the necessary data to train the models. By commenting out the line:

list(product(*[l1, l2, l3, activ])), the parametric search is deactivated, and only the architecture specified in the next line is fitted

Additionally, some timing features are implemented. Passing a large parametric space can result in the program running for over 4 hours, training each model. Because of this, a progress meter is added so that the user can see how far along the program is.

[8]:
```
models = []
aic_scores = []
r2_scores = []
maes = []
rmses = []
l1 = np.linspace(32, 256, 5)
l2 = np.linspace(0, 256, 5)
l3 = np.linspace(0, 256, 5)
activ = ['relu', 'tanh']
# parametric_space = list(product(*[l1, l2, l3, activ]))
parametric_space = [[200, 64, 192, 'relu']]
print(parametric_space)
num_hidden = len(list(i for i in parametric_space[0] if isinstance(i, (int or␣
↪float))))
num_hyper = len(parametric_space[0]) - num_hidden
start_t = time.time()
c = 1
```

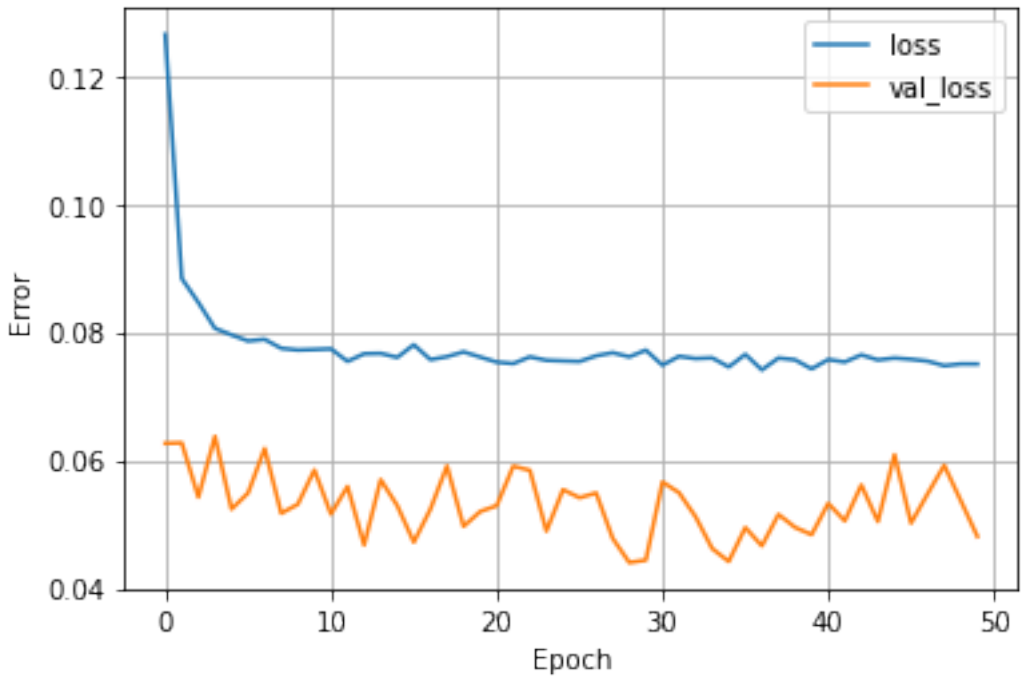
```
[[200, 64, 192, 'relu']]
```

0.7 5) Running the model training

This for loop passes each architecture specified in the parametric space into the build_and_compile_model() function and reports the accuracy metrics into their specific list. It also updates the progress each time a model is finished evaluation

[9]:
```
for arch in parametric_space:
    print('Progress: ' + str(c) + '/' + str(len(parametric_space)))
    dnn_model, aic, r2, mae, rmse, test_predictions = fit_and_evaluate(arch)
    models.append(dnn_model)
    aic_scores.append(aic)
    r2_scores.append(r2)
    maes.append(mae)
    rmses.append(rmse)

    curr_time = time.time()
    diff_t = curr_time - start_t
    t_per_model = diff_t / c
    num_mods_rem = len(parametric_space) - c
    t_rem = t_per_model * num_mods_rem
    print("Estimated Time Remaining: " + time.strftime('%H:%M:%S', time.
↪gmtime(t_rem)) + ' seconds')
    c += 1
```

```
Progress: 1/1
```




```
Estimated Time Remaining: 00:00:00 seconds
```

0.8 6) Choosing the best model

After the parametric seach is finished, the accuracy metrics are compiled into a csv file. The user must look through these results and pick the model which has the LOWEST AIC score and HIGHEST R-Squared.

The final 2 lines save the zeroth model in the parametric space for later use. During the parametric seach, this should be disabled because the zeroth model is likely not the most accurate. When the parametric search is disabled (only a single architecture is being fitted) this can be re-enabled to save the model.

[10]:
```
parametric_space_t = np.asarray(parametric_space).transpose().tolist()
output_data = [parametric_space_t[0], parametric_space_t[1],␣
 ↪parametric_space_t[2], aic_scores, maes, rmses, r2_scores]
output_data = np.asarray(output_data).transpose().tolist()
print(output_data)
oput = pd.DataFrame(output_data, columns=['L1', 'L2', 'L3', 'AIC', 'MAE',␣
 ↪'RMSE', 'R2'])
# print(oput)
# oput.to_csv('Parametric_space_study.csv', index=False)
print(models[0].summary())
out_path = str(cwd.parent) + '/models/trial0.3.h5'
# models[0].save(out_path)
```

```
[['200', '64', '192', '-42249.02537623041', '0.04866076749821012',
'0.21317433109464726', '0.8131798652629634']]
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 6)]               0

 batch_normalization (BatchN  (None, 6)                24
 ormalization)

 dense (Dense)               (None, 200)               1400

 dropout (Dropout)           (None, 200)               0

 dense_1 (Dense)             (None, 64)                12864

 dropout_1 (Dropout)         (None, 64)                0

 dense_2 (Dense)             (None, 192)               12480

 dropout_2 (Dropout)         (None, 192)               0

 dense_3 (Dense)             (None, 1)                 193

=================================================================
Total params: 26,961
Trainable params: 26,949
Non-trainable params: 12
_________________________________________________________________
None
```

0.9 HW 9: Results & Conclusions

0.9.1 OCEN 460

0.9.2 Team: _/Sample_Text/

0.9.3 Members: Nate Baker and James Frizzell

Overview:

The purpose of this project is to use existing data on the growth of coral to predict whether coral can grow given oceanographic conditions. The latitude, longitude, depth, temperature, salinity, and dissolved oxygen levels are used to predict a binary value with 1 meaning that coral can grow and 0 meaning that coral cannot grow. This value will then be multiplied by a factor of 100 to represent a percentage.

1 1) Setup Class Structure

This code has been written in an object-oriented fashion so that it may easily be packaged and shipped into production later.

First, we set up the class structure and import the internal libraries used.

[24]:
```
import random

import tensorflow as tf
import numpy as np
from itertools import product
from statistics import mean, median
```

[25]:
```
class CoralPrediction:
    def __init__(self):
        self.model = None
        self.params = np.zeros(6)
        self.test_lat = []
        self.test_long = []
        self.test_depth = []
        self.test_temp = []
        self.test_sal = []
        self.test_oxy = []

    def set_model(self, modelpath):
        self.model = tf.keras.models.load_model(modelpath)

    def predict(self, params):
        self.params = params
        missing = []
        for i in range(len(self.params)):
            if type(self.params[i]) != float and type(self.params[i]) != int:
                missing.append(i)

        if 0 in missing:
            self.test_lat = np.linspace(-90, 90, 90)
        else:
            self.test_lat = [self.params[0]]

        if 1 in missing:
            self.test_long = np.linspace(-180, 180, 180)
        else:
            self.test_long = [self.params[1]]

        if 2 in missing:
            self.test_depth = np.linspace(0, 3000, 50)
        else:
```

```
            self.test_depth = [self.params[2]]

        if 3 in missing:
            self.test_temp = np.linspace(-2, 28, 20)
        else:
            self.test_temp = [self.params[3]]

        if 4 in missing:
            self.test_sal = np.linspace(0, 41, 20)
        else:
            self.test_sal = [self.params[4]]

        if 5 in missing:
            self.test_oxy = np.linspace(0.2, 132, 40)
        else:
            self.test_oxy = [self.params[5]]

        cond_list = list(product(*[self.test_lat, self.test_long, self.
↪test_depth, self.test_temp, self.test_sal, self.test_oxy]))
        if len(cond_list) >= 4000:
            cond_list = random.sample(cond_list, 4000)
        count = 0
        predictions = []
        for cond in cond_list:
            print(count, ' completed out of: ', len(cond_list))
            predictions.append(self.model.predict([list(cond)])[0][0])
            count += 1
        return mean(predictions), median(predictions)
```

The CoralPrediction class has a few properties and methods. Once the object has been instantiated in the main.py function, the user will assign a trained tensorflow model to the predictor using the set_model function.

Then, the user will pass the sea state conditions into the predictor model with the predict function. The code will evaluate the conditions and return the probability of coral being able to grow given the inputs.

Some precautions have been implemented to assist the user in the case that not all metocean data is known. This will be discussed further in section 3.

2 2) MAIN.py, Runnable function

[26]:
```
import os
import pathlib
# import CoralClass


def __main__(modelpath, conditions):
```

```
    CoralPredictor = CoralPrediction()
    CoralPredictor.set_model(modelpath)
    mean_pred, med_pred = CoralPredictor.predict(conditions)
    print('\n----------------------------------------')
    print('The average likelihood of coral growth is: ', mean_pred*100, '%')
    print('The median likelihood of coral growth is: ', med_pred*100, '%')


# if __name__ == '__main__':           Failed implementation of command-line␣
 ↪run functionality
#     import sys
#     args = sys.argv[2:]
#     modelpath = sys.argv[1]
#     pythonname = sys.argv[0]
#     __main__(modelpath, args)

# change this to change the model that will be used to predict the coral growth
modelpath = str(pathlib.Path(os.getcwd()).parent) + '/coral-prediction/models/
 ↪trial0.3.h5'

# Enter the metocean conditions here   [Latitude, Longitude, Depth (m),␣
 ↪Temperature (C), Salinity (ppt), Dissolved O2 (umol/kg)]
# Enter "None" if the data is unknown
conditions = [20, -150, 1000, None, 30, None]

__main__(modelpath, conditions)
```

```
0  completed out of:  800
1  completed out of:  800
2  completed out of:  800
3  completed out of:  800
4  completed out of:  800
5  completed out of:  800
6  completed out of:  800
7  completed out of:  800
8  completed out of:  800
9  completed out of:  800
10  completed out of:  800
11  completed out of:  800
12  completed out of:  800
13  completed out of:  800
14  completed out of:  800
15  completed out of:  800
16  completed out of:  800
17  completed out of:  800
18  completed out of:  800
19  completed out of:  800
```

```
20  completed out of:  800
21  completed out of:  800
22  completed out of:  800
23  completed out of:  800
24  completed out of:  800
25  completed out of:  800
26  completed out of:  800
27  completed out of:  800
28  completed out of:  800
29  completed out of:  800
30  completed out of:  800
31  completed out of:  800
32  completed out of:  800
33  completed out of:  800
34  completed out of:  800
35  completed out of:  800
36  completed out of:  800
37  completed out of:  800
38  completed out of:  800
39  completed out of:  800
40  completed out of:  800
41  completed out of:  800
42  completed out of:  800
43  completed out of:  800
44  completed out of:  800
45  completed out of:  800
46  completed out of:  800
47  completed out of:  800
48  completed out of:  800
49  completed out of:  800
50  completed out of:  800
51  completed out of:  800
52  completed out of:  800
53  completed out of:  800
54  completed out of:  800
55  completed out of:  800
56  completed out of:  800
57  completed out of:  800
58  completed out of:  800
59  completed out of:  800
60  completed out of:  800
61  completed out of:  800
62  completed out of:  800
63  completed out of:  800
64  completed out of:  800
65  completed out of:  800
66  completed out of:  800
67  completed out of:  800
```

```
68  completed out of:  800
69  completed out of:  800
70  completed out of:  800
71  completed out of:  800
72  completed out of:  800
73  completed out of:  800
74  completed out of:  800
75  completed out of:  800
76  completed out of:  800
77  completed out of:  800
78  completed out of:  800
79  completed out of:  800
80  completed out of:  800
81  completed out of:  800
82  completed out of:  800
83  completed out of:  800
84  completed out of:  800
85  completed out of:  800
86  completed out of:  800
87  completed out of:  800
88  completed out of:  800
89  completed out of:  800
90  completed out of:  800
91  completed out of:  800
92  completed out of:  800
93  completed out of:  800
94  completed out of:  800
95  completed out of:  800
96  completed out of:  800
97  completed out of:  800
98  completed out of:  800
99  completed out of:  800
100  completed out of:  800
101  completed out of:  800
102  completed out of:  800
103  completed out of:  800
104  completed out of:  800
105  completed out of:  800
106  completed out of:  800
107  completed out of:  800
108  completed out of:  800
109  completed out of:  800
110  completed out of:  800
111  completed out of:  800
112  completed out of:  800
113  completed out of:  800
114  completed out of:  800
115  completed out of:  800
```

```
116  completed out of:  800
117  completed out of:  800
118  completed out of:  800
119  completed out of:  800
120  completed out of:  800
121  completed out of:  800
122  completed out of:  800
123  completed out of:  800
124  completed out of:  800
125  completed out of:  800
126  completed out of:  800
127  completed out of:  800
128  completed out of:  800
129  completed out of:  800
130  completed out of:  800
131  completed out of:  800
132  completed out of:  800
133  completed out of:  800
134  completed out of:  800
135  completed out of:  800
136  completed out of:  800
137  completed out of:  800
138  completed out of:  800
139  completed out of:  800
140  completed out of:  800
141  completed out of:  800
142  completed out of:  800
143  completed out of:  800
144  completed out of:  800
145  completed out of:  800
146  completed out of:  800
147  completed out of:  800
148  completed out of:  800
149  completed out of:  800
150  completed out of:  800
151  completed out of:  800
152  completed out of:  800
153  completed out of:  800
154  completed out of:  800
155  completed out of:  800
156  completed out of:  800
157  completed out of:  800
158  completed out of:  800
159  completed out of:  800
160  completed out of:  800
161  completed out of:  800
162  completed out of:  800
163  completed out of:  800
```

```
164  completed out of:  800
165  completed out of:  800
166  completed out of:  800
167  completed out of:  800
168  completed out of:  800
169  completed out of:  800
170  completed out of:  800
171  completed out of:  800
172  completed out of:  800
173  completed out of:  800
174  completed out of:  800
175  completed out of:  800
176  completed out of:  800
177  completed out of:  800
178  completed out of:  800
179  completed out of:  800
180  completed out of:  800
181  completed out of:  800
182  completed out of:  800
183  completed out of:  800
184  completed out of:  800
185  completed out of:  800
186  completed out of:  800
187  completed out of:  800
188  completed out of:  800
189  completed out of:  800
190  completed out of:  800
191  completed out of:  800
192  completed out of:  800
193  completed out of:  800
194  completed out of:  800
195  completed out of:  800
196  completed out of:  800
197  completed out of:  800
198  completed out of:  800
199  completed out of:  800
200  completed out of:  800
201  completed out of:  800
202  completed out of:  800
203  completed out of:  800
204  completed out of:  800
205  completed out of:  800
206  completed out of:  800
207  completed out of:  800
208  completed out of:  800
209  completed out of:  800
210  completed out of:  800
211  completed out of:  800
```

```
212  completed out of:  800
213  completed out of:  800
214  completed out of:  800
215  completed out of:  800
216  completed out of:  800
217  completed out of:  800
218  completed out of:  800
219  completed out of:  800
220  completed out of:  800
221  completed out of:  800
222  completed out of:  800
223  completed out of:  800
224  completed out of:  800
225  completed out of:  800
226  completed out of:  800
227  completed out of:  800
228  completed out of:  800
229  completed out of:  800
230  completed out of:  800
231  completed out of:  800
232  completed out of:  800
233  completed out of:  800
234  completed out of:  800
235  completed out of:  800
236  completed out of:  800
237  completed out of:  800
238  completed out of:  800
239  completed out of:  800
240  completed out of:  800
241  completed out of:  800
242  completed out of:  800
243  completed out of:  800
244  completed out of:  800
245  completed out of:  800
246  completed out of:  800
247  completed out of:  800
248  completed out of:  800
249  completed out of:  800
250  completed out of:  800
251  completed out of:  800
252  completed out of:  800
253  completed out of:  800
254  completed out of:  800
255  completed out of:  800
256  completed out of:  800
257  completed out of:  800
258  completed out of:  800
259  completed out of:  800
```

```
260  completed out of:  800
261  completed out of:  800
262  completed out of:  800
263  completed out of:  800
264  completed out of:  800
265  completed out of:  800
266  completed out of:  800
267  completed out of:  800
268  completed out of:  800
269  completed out of:  800
270  completed out of:  800
271  completed out of:  800
272  completed out of:  800
273  completed out of:  800
274  completed out of:  800
275  completed out of:  800
276  completed out of:  800
277  completed out of:  800
278  completed out of:  800
279  completed out of:  800
280  completed out of:  800
281  completed out of:  800
282  completed out of:  800
283  completed out of:  800
284  completed out of:  800
285  completed out of:  800
286  completed out of:  800
287  completed out of:  800
288  completed out of:  800
289  completed out of:  800
290  completed out of:  800
291  completed out of:  800
292  completed out of:  800
293  completed out of:  800
294  completed out of:  800
295  completed out of:  800
296  completed out of:  800
297  completed out of:  800
298  completed out of:  800
299  completed out of:  800
300  completed out of:  800
301  completed out of:  800
302  completed out of:  800
303  completed out of:  800
304  completed out of:  800
305  completed out of:  800
306  completed out of:  800
307  completed out of:  800
```

```
308  completed out of:  800
309  completed out of:  800
310  completed out of:  800
311  completed out of:  800
312  completed out of:  800
313  completed out of:  800
314  completed out of:  800
315  completed out of:  800
316  completed out of:  800
317  completed out of:  800
318  completed out of:  800
319  completed out of:  800
320  completed out of:  800
321  completed out of:  800
322  completed out of:  800
323  completed out of:  800
324  completed out of:  800
325  completed out of:  800
326  completed out of:  800
327  completed out of:  800
328  completed out of:  800
329  completed out of:  800
330  completed out of:  800
331  completed out of:  800
332  completed out of:  800
333  completed out of:  800
334  completed out of:  800
335  completed out of:  800
336  completed out of:  800
337  completed out of:  800
338  completed out of:  800
339  completed out of:  800
340  completed out of:  800
341  completed out of:  800
342  completed out of:  800
343  completed out of:  800
344  completed out of:  800
345  completed out of:  800
346  completed out of:  800
347  completed out of:  800
348  completed out of:  800
349  completed out of:  800
350  completed out of:  800
351  completed out of:  800
352  completed out of:  800
353  completed out of:  800
354  completed out of:  800
355  completed out of:  800
```

```
356  completed out of:  800
357  completed out of:  800
358  completed out of:  800
359  completed out of:  800
360  completed out of:  800
361  completed out of:  800
362  completed out of:  800
363  completed out of:  800
364  completed out of:  800
365  completed out of:  800
366  completed out of:  800
367  completed out of:  800
368  completed out of:  800
369  completed out of:  800
370  completed out of:  800
371  completed out of:  800
372  completed out of:  800
373  completed out of:  800
374  completed out of:  800
375  completed out of:  800
376  completed out of:  800
377  completed out of:  800
378  completed out of:  800
379  completed out of:  800
380  completed out of:  800
381  completed out of:  800
382  completed out of:  800
383  completed out of:  800
384  completed out of:  800
385  completed out of:  800
386  completed out of:  800
387  completed out of:  800
388  completed out of:  800
389  completed out of:  800
390  completed out of:  800
391  completed out of:  800
392  completed out of:  800
393  completed out of:  800
394  completed out of:  800
395  completed out of:  800
396  completed out of:  800
397  completed out of:  800
398  completed out of:  800
399  completed out of:  800
400  completed out of:  800
401  completed out of:  800
402  completed out of:  800
403  completed out of:  800
```

```
404  completed out of:  800
405  completed out of:  800
406  completed out of:  800
407  completed out of:  800
408  completed out of:  800
409  completed out of:  800
410  completed out of:  800
411  completed out of:  800
412  completed out of:  800
413  completed out of:  800
414  completed out of:  800
415  completed out of:  800
416  completed out of:  800
417  completed out of:  800
418  completed out of:  800
419  completed out of:  800
420  completed out of:  800
421  completed out of:  800
422  completed out of:  800
423  completed out of:  800
424  completed out of:  800
425  completed out of:  800
426  completed out of:  800
427  completed out of:  800
428  completed out of:  800
429  completed out of:  800
430  completed out of:  800
431  completed out of:  800
432  completed out of:  800
433  completed out of:  800
434  completed out of:  800
435  completed out of:  800
436  completed out of:  800
437  completed out of:  800
438  completed out of:  800
439  completed out of:  800
440  completed out of:  800
441  completed out of:  800
442  completed out of:  800
443  completed out of:  800
444  completed out of:  800
445  completed out of:  800
446  completed out of:  800
447  completed out of:  800
448  completed out of:  800
449  completed out of:  800
450  completed out of:  800
451  completed out of:  800
```

```
452  completed out of:  800
453  completed out of:  800
454  completed out of:  800
455  completed out of:  800
456  completed out of:  800
457  completed out of:  800
458  completed out of:  800
459  completed out of:  800
460  completed out of:  800
461  completed out of:  800
462  completed out of:  800
463  completed out of:  800
464  completed out of:  800
465  completed out of:  800
466  completed out of:  800
467  completed out of:  800
468  completed out of:  800
469  completed out of:  800
470  completed out of:  800
471  completed out of:  800
472  completed out of:  800
473  completed out of:  800
474  completed out of:  800
475  completed out of:  800
476  completed out of:  800
477  completed out of:  800
478  completed out of:  800
479  completed out of:  800
480  completed out of:  800
481  completed out of:  800
482  completed out of:  800
483  completed out of:  800
484  completed out of:  800
485  completed out of:  800
486  completed out of:  800
487  completed out of:  800
488  completed out of:  800
489  completed out of:  800
490  completed out of:  800
491  completed out of:  800
492  completed out of:  800
493  completed out of:  800
494  completed out of:  800
495  completed out of:  800
496  completed out of:  800
497  completed out of:  800
498  completed out of:  800
499  completed out of:  800
```

```
500  completed out of:  800
501  completed out of:  800
502  completed out of:  800
503  completed out of:  800
504  completed out of:  800
505  completed out of:  800
506  completed out of:  800
507  completed out of:  800
508  completed out of:  800
509  completed out of:  800
510  completed out of:  800
511  completed out of:  800
512  completed out of:  800
513  completed out of:  800
514  completed out of:  800
515  completed out of:  800
516  completed out of:  800
517  completed out of:  800
518  completed out of:  800
519  completed out of:  800
520  completed out of:  800
521  completed out of:  800
522  completed out of:  800
523  completed out of:  800
524  completed out of:  800
525  completed out of:  800
526  completed out of:  800
527  completed out of:  800
528  completed out of:  800
529  completed out of:  800
530  completed out of:  800
531  completed out of:  800
532  completed out of:  800
533  completed out of:  800
534  completed out of:  800
535  completed out of:  800
536  completed out of:  800
537  completed out of:  800
538  completed out of:  800
539  completed out of:  800
540  completed out of:  800
541  completed out of:  800
542  completed out of:  800
543  completed out of:  800
544  completed out of:  800
545  completed out of:  800
546  completed out of:  800
547  completed out of:  800
```

```
548  completed out of:  800
549  completed out of:  800
550  completed out of:  800
551  completed out of:  800
552  completed out of:  800
553  completed out of:  800
554  completed out of:  800
555  completed out of:  800
556  completed out of:  800
557  completed out of:  800
558  completed out of:  800
559  completed out of:  800
560  completed out of:  800
561  completed out of:  800
562  completed out of:  800
563  completed out of:  800
564  completed out of:  800
565  completed out of:  800
566  completed out of:  800
567  completed out of:  800
568  completed out of:  800
569  completed out of:  800
570  completed out of:  800
571  completed out of:  800
572  completed out of:  800
573  completed out of:  800
574  completed out of:  800
575  completed out of:  800
576  completed out of:  800
577  completed out of:  800
578  completed out of:  800
579  completed out of:  800
580  completed out of:  800
581  completed out of:  800
582  completed out of:  800
583  completed out of:  800
584  completed out of:  800
585  completed out of:  800
586  completed out of:  800
587  completed out of:  800
588  completed out of:  800
589  completed out of:  800
590  completed out of:  800
591  completed out of:  800
592  completed out of:  800
593  completed out of:  800
594  completed out of:  800
595  completed out of:  800
```

```
596  completed out of:  800
597  completed out of:  800
598  completed out of:  800
599  completed out of:  800
600  completed out of:  800
601  completed out of:  800
602  completed out of:  800
603  completed out of:  800
604  completed out of:  800
605  completed out of:  800
606  completed out of:  800
607  completed out of:  800
608  completed out of:  800
609  completed out of:  800
610  completed out of:  800
611  completed out of:  800
612  completed out of:  800
613  completed out of:  800
614  completed out of:  800
615  completed out of:  800
616  completed out of:  800
617  completed out of:  800
618  completed out of:  800
619  completed out of:  800
620  completed out of:  800
621  completed out of:  800
622  completed out of:  800
623  completed out of:  800
624  completed out of:  800
625  completed out of:  800
626  completed out of:  800
627  completed out of:  800
628  completed out of:  800
629  completed out of:  800
630  completed out of:  800
631  completed out of:  800
632  completed out of:  800
633  completed out of:  800
634  completed out of:  800
635  completed out of:  800
636  completed out of:  800
637  completed out of:  800
638  completed out of:  800
639  completed out of:  800
640  completed out of:  800
641  completed out of:  800
642  completed out of:  800
643  completed out of:  800
```

```
644  completed out of:  800
645  completed out of:  800
646  completed out of:  800
647  completed out of:  800
648  completed out of:  800
649  completed out of:  800
650  completed out of:  800
651  completed out of:  800
652  completed out of:  800
653  completed out of:  800
654  completed out of:  800
655  completed out of:  800
656  completed out of:  800
657  completed out of:  800
658  completed out of:  800
659  completed out of:  800
660  completed out of:  800
661  completed out of:  800
662  completed out of:  800
663  completed out of:  800
664  completed out of:  800
665  completed out of:  800
666  completed out of:  800
667  completed out of:  800
668  completed out of:  800
669  completed out of:  800
670  completed out of:  800
671  completed out of:  800
672  completed out of:  800
673  completed out of:  800
674  completed out of:  800
675  completed out of:  800
676  completed out of:  800
677  completed out of:  800
678  completed out of:  800
679  completed out of:  800
680  completed out of:  800
681  completed out of:  800
682  completed out of:  800
683  completed out of:  800
684  completed out of:  800
685  completed out of:  800
686  completed out of:  800
687  completed out of:  800
688  completed out of:  800
689  completed out of:  800
690  completed out of:  800
691  completed out of:  800
```

```
692  completed out of:  800
693  completed out of:  800
694  completed out of:  800
695  completed out of:  800
696  completed out of:  800
697  completed out of:  800
698  completed out of:  800
699  completed out of:  800
700  completed out of:  800
701  completed out of:  800
702  completed out of:  800
703  completed out of:  800
704  completed out of:  800
705  completed out of:  800
706  completed out of:  800
707  completed out of:  800
708  completed out of:  800
709  completed out of:  800
710  completed out of:  800
711  completed out of:  800
712  completed out of:  800
713  completed out of:  800
714  completed out of:  800
715  completed out of:  800
716  completed out of:  800
717  completed out of:  800
718  completed out of:  800
719  completed out of:  800
720  completed out of:  800
721  completed out of:  800
722  completed out of:  800
723  completed out of:  800
724  completed out of:  800
725  completed out of:  800
726  completed out of:  800
727  completed out of:  800
728  completed out of:  800
729  completed out of:  800
730  completed out of:  800
731  completed out of:  800
732  completed out of:  800
733  completed out of:  800
734  completed out of:  800
735  completed out of:  800
736  completed out of:  800
737  completed out of:  800
738  completed out of:  800
739  completed out of:  800
```

```
740  completed out of:  800
741  completed out of:  800
742  completed out of:  800
743  completed out of:  800
744  completed out of:  800
745  completed out of:  800
746  completed out of:  800
747  completed out of:  800
748  completed out of:  800
749  completed out of:  800
750  completed out of:  800
751  completed out of:  800
752  completed out of:  800
753  completed out of:  800
754  completed out of:  800
755  completed out of:  800
756  completed out of:  800
757  completed out of:  800
758  completed out of:  800
759  completed out of:  800
760  completed out of:  800
761  completed out of:  800
762  completed out of:  800
763  completed out of:  800
764  completed out of:  800
765  completed out of:  800
766  completed out of:  800
767  completed out of:  800
768  completed out of:  800
769  completed out of:  800
770  completed out of:  800
771  completed out of:  800
772  completed out of:  800
773  completed out of:  800
774  completed out of:  800
775  completed out of:  800
776  completed out of:  800
777  completed out of:  800
778  completed out of:  800
779  completed out of:  800
780  completed out of:  800
781  completed out of:  800
782  completed out of:  800
783  completed out of:  800
784  completed out of:  800
785  completed out of:  800
786  completed out of:  800
787  completed out of:  800
```

```
788  completed out of:  800
789  completed out of:  800
790  completed out of:  800
791  completed out of:  800
792  completed out of:  800
793  completed out of:  800
794  completed out of:  800
795  completed out of:  800
796  completed out of:  800
797  completed out of:  800
798  completed out of:  800
799  completed out of:  800

----------------------------------------
The average likelihood of coral growth is:  63.69994878768921 %
The median likelihood of coral growth is:  98.32048416137695 %
```

This code allows the user to input conditions and see the output. First, the user must specify the path of a trained tensorflow model which will be used to predict the growth of the coral.

Then, the user inputs metocean conditions and runs the program.

The code will report the mean and median likelihood that coral can grown in the given conditions. If all the parameters are specified, the mean will equal the median (since there is only 1 data point that is predicted)

3 3) Probablistic Modelling

In many real-world cases, not all the data will be known for the location of interest. For example, a climatologist may be studying the South Pacific for coral growth around the Great Barrier Reef. The scientist will have a GPS coordinate of interest (latitude and longitude) and likely will know the ocean depth at that location. However, the temperature, salinity and dissolved oxygen content may be unknown. How can the research continue if the input data is incomplete?

This is solved with probabilistic prediction. In the case that some inputs are not known (are entered as "None" in the inputs) the code will sweep through a range of possible values based on the training data that was earlier used. For each value in the range, the program will predict the coral growth probability. Once all the possible values have been simulated, the code will take the average and median probability and report it back to the user.

4 4) Conclusions

This project has been successful in predicting the ability of coral to grow in certain metocean conditions by merging data from multiple sources. In the future, a higher resolution dataset could improve the accuracy of the model further.