1. Comparing Machine Learning Models for Predicting Chlorophyll Concentrations: A Thesis Introduction Paper

Jordan Reichhardt

2. Abstract/Summary

Chlorophyll-a concentrations are a direct proxy for the phytoplankton levels in various ocean environments worldwide. In other words chlorophyll-a levels serve as a direct indicator of marine ecosystem health. Historically in-situ chlorophyll measurements have provided accurate data but spacial and temporal limitations. Additionally, satellite measurements provide global coverage, but perform poorly in coastal environments. This study uses multiple machine learning methods along with a combination of in-situ measurements, environmental, swell, and satellite forcing features to estimate chlorophyll-a concentrations at the Cal Poly pier (CPXC1). The study utilizes five years (from 2020 to 2024) of in situ chlorophyll measurements from the Cal Poly pier (CPXC1) in Port San Luis, along with time-aligned environmental, swell, and satellite data. The two machine learning approaches compared were eXtreme Gradient Boosting and Feedforward Neural Networks (FNNs). These were trained and tested using an 80/20 randomized training and testing split. The model is analyzed based on the coefficient of determination (R^2), mean absolute error, root mean squared error (RMSE), and prediction bias. The results showed that the XGBoost decision tree model achieved a testing R^2 of 0.85 which outperformed the FNN model, which achieved a testing R^2 of 0.69. Both models underpredicted large chlorophyll peaks. Additionally, analysis of feature importance indicated that satellite-derived chlorophyll metrics and swell variables correlated the most strongly with fluctuations in chlorophyll concentrations. In this research, tree-based methods are well-suited for modeling chlorophyll concentrations in complex coastal environments. This method and similar models may prove useful in improving global models of coastal water quality monitoring using integrated environmental and swell datasets.

3. Problem Statement

Monitoring Coastal is a primary indicator for marine ecosystem health, harmful algal bloom (HAB) detection, fishery production, and recreational activities. Chlorophyll-a concentration highly correlates with phytoplankton concentration in a given volume of water. Historically, chlorophyll measurements have been done using in-situ sampling. These provide accurate measurements but are limited in their spatial and temporal abundance.

More recently, Satellites have been used to estimate chlorophyll levels. These estimates, however, perform poorly in coastal waters. Nearshore environments have many variables that impact the chlorophyll levels. These variables include wave-driven mixing, runoff, atmospheric interference, and bottom reflectance. All of these factors can make satellite estimates less accurate. Accordingly, alternative approaches that include the impacts of environmental conditions nearshore must be used to improve chlorophyll predictions within dynamic coastal environments.

This project analyzes and estimates the relationship between environmental conditions and coastal chlorophyll concentrations off the Cal Poly pier (NOAA station CPXC1). The data is comprised of 5 years of in-situ chlorophyll measurements taken every 30 minutes. These in-situ (ground-truth) values are time-aligned with environmental features across the five-year timespan. The environmental features include swell conditions, atmospheric conditions, weather surrounding the Cal Poly pier, and Sentinel satellite estimates for the shore off the Cal Poly pier. These data types are used as inputs to multiple machine learning models and trained in conjunction with the ground truth variables to derive relationships between chlorophyll values and environmental forcing features.

The research compares the performance of Extreme Gradient Boosting (XGBoost) and Feedforward Neural Networks (FNN) machine learning models. The models are assessed with widely used regression metrics such as the Coefficient of determination(R^2) for both training and testing, mean absolute error (MAE), root mean squared error (RMSE), and prediction bias. In comparing the two machine learning approaches, the research aims to analyze the strengths and limitations of the tree-based ML method and neural network ML method, which are two commonly used models for environmental datasets. Ultimately, the research aims to determine the most fitting model for determining the relationship between environmental features and chlorophyll in a complex coastal environment.

4. Literature Review

Qing et al. (2021)

Qing et al. [1] in their paper "*Improving remote sensing retrieval of water clarity in complex coastal and inland waters with modified absorption estimation and optical water classification using Sentinel-2 MSI*" investigate the water quality parameters of inland lakes in various coastal lakes and seas in China using machine learning methods for satellite water quality estimates. The author explores the unique optical properties of various coastal environments and creates individual models for each location. The study is advantageous to the development of this chlorophyll research as it focuses on water quality prediction, including chlorophyll predictions in optically complex environments. Qing's study, however, does not include environmental forcing features in its predictions of water quality. Ultimately, this gap motivates further research which includes environmental and swell forcing features in predicting chlorophyll levels.

Brockman et al. (2016)

Brockmann et al. [2] in "Evolution of the C2RCC Neural Network for Sentinel-2 and Sentinel-3 for the Retrieval of Ocean Colour Products in Normal and Extreme Optically Complex Waters," test updated methods to predict water quality using neural network methods for processing Sentinel-2 and Sentinel-3 satellite imagery. The model uses Case 2 Regional CoastColour (C2RCC), which is a standardized neural network originally developed through the European CoastColour project that is now used for Sentinel-2 multispectral imagery and Sentinel-3 OLCI processing. This method is advantageous because it models non-linear relationships between spectral reflectance and water quality. The paper uses data from coastal, estuary, and inland lakes to test an updated version of C2RCC. They then validated the predictions against in-situ measurements and showed improvements in chlorophyll predictions with the updated predictive

model. This paper supports the ability of Neural Networks to model non-linear optical relationships, particularly those derived from satellites.

Ali et al. (2024)

Deep learning techniques that estimate water quality are designed, tested, and analyzed in Ali et al.'s "*Deep learning for water quality multivariate assessment in inland water across China*" [3]. This research focuses on estimating water quality parameters, including chlorophyll levels in various inland lakes in China. The researchers use Sentinel-2 imagery for the lakes under study to derive their own chlorophyll optical predictions from raw wavelength values. The blue, red, and green wavelengths are used as inputs to multiple machine learning models, including Deep Neural Networks, Support Vector Machines, Random Forests, and XGBoost models. This study only uses satellite data as inputs to the coastal predictions of chlorophyll. The study reinforces the need for the inclusion of environmental and swell data in models that estimate chlorophyll. Additionally, the study provides evidence that neural networks are most effective for chlorophyll estimations, yet all predictive models underestimate chlorophyll, particularly the peaks.

Yao et al. (2023)

Yao et al. [4] develop models to predict future chlorophyll-a satellite retrieval concentrations using remote sensing satellite data in their paper "Prediction of Sea Surface Chlorophyll-a Concentrations Based on Deep Learning and Time-Series Remote Sensing Data". The researchers analyze the effectiveness of multiple predictive architectures, including ConvLSTM and SA-ConvLSTM, in predicting future satellite chlorophyll concentrations in the Yellow Sea and Bohai Sea using environmental forcing features and satellite data. Yao's Research is most similar to the research explored in this paper, but varies spatially and temporally. The group is predicting a monthly chlorophyll average for the entire Yellow Sea and the entire Bohai Sea. While their time split testing and training results indicate strong predictive power, the predictions are highly averaged over space and time.

Chen and Guestrin (2016)

Chen and Guestrin [5] introduced in their paper "XGBoost, a Scalable Tree Boosting System" a gradient boosted decision tree model. This algorithm is designed for consecutive corrections to errors produced in each previous tree. The algorithm includes regularization methods that reduce the tendency of the model to overfit, particularly for large models. XGBoost decision trees excel at modeling nonlinear datasets, which are particularly applicable for environmental applications. The researcher notes that one limitation of this method is that it depends on hyperparameter tuning, meaning the number of trees, depth of trees, and regularization, among other parameters, must be optimized in order for the model to work effectively. This research forms the foundation of the XGBoost decision trees model used in this paper.

LeCun et al. (2015)

LeCun et al. [6] detail a review of multiple deep learning methods in their paper "*Deep Learning*". Their work focuses on Neural Networks with multiple hidden layers, which can learn relationships between complex nonlinear data. The paper presents the success of deep learning methods in applications such as image processing, speech recognition, and scientific studies. The paper analyzes Feedforward Neural Networks (FNNs), Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Long Short-Term Memory Networks (LSTMs), Encoder-Decoder Networks, and Memory-Augmented Networks. Of these methods, the FNNs and Short-Term Memory networks are the most relevant to the research in this paper, as FNNs are implemented and LSTMs are a method to consider in future work.

O'Reilly et al. (1998)

O'Reilly et al. [7] in their paper "*Ocean Color Chlorophyll Algorithms for SeaWiFS*" developed methods to estimate chlorophyll-a concentrations from ocean color reflectance. The group used large in-situ measurement datasets to derive coefficients that match satellite wavelengths to chlorophyll values. The research develops the OCx family of algorithms which estimates chlorophyll concentrations from red, green, and blue wavelengths. These algorithms are displayed in table 4.1. Here, Chl (chlorophyll-a concentration) is derived from R (band ratio) a_s (empirical coefficients), Rrs(λ) (remote sensing reflectance, Rrs(443) (reflectance at 443 nm), Rrs(490) (reflectance at 490 nm), Rrs(510) (reflectance at 510 nm), and Rrs(555) (reflectance at 555 nm). These methods, however, focus on open ocean conditions and are ineffective in predicting coastal chlorophyll levels.

Algorithm	Equation	Band Ratio
OC2	$\mathrm{Chl} = 10^{a_0+a_{1R}+a_{2R}^2+a_{3R}^3+a_{4R}^4}$	$\mathrm{R} = \log_{10}\frac{R_{rs(490)}}{R_{rs(555)}}$
OC3	$\mathrm{Chl} = 10^{a_0+a_{1R}+a_{2R}^2+a_{3R}^3+a_{4R}^4}$	$\mathrm{R} = \log_{10}\left(\frac{\max(R_{rs(443)}, R_{rs(490)})}{R_{rs(555)}}\right)$
OC4	$\mathrm{Chl} = 10^{a_0+a_{1R}+a_{2R}^2+a_{3R}^3+a_{4R}^4}$	$\mathrm{R} = \log_{10}\left(\frac{\max(R_{rs(443)}, R_{rs(490)}, R_{rs(510)})}{R_{rs(555)}}\right)$

Table 4.1: O'Reilly et al. derived OCx algorithms for remote chlorophyll retrieval.

5. Your Approach and/or Algorithm

5.0 High Level Model Overview

The research compares multiple regressive Machine Learning (ML) models to predict chlorophyll values and using various environmental, swell, and satellite inputs. The inputs and the in-situ values are collected from the Cal Poly pier and time-aligned in 30 minute intervals so that they can be organized and input into feature matrices. These inputs and ground truth values are then used in both XGBoost regression and FNN models in an 80/20 training and testing split.

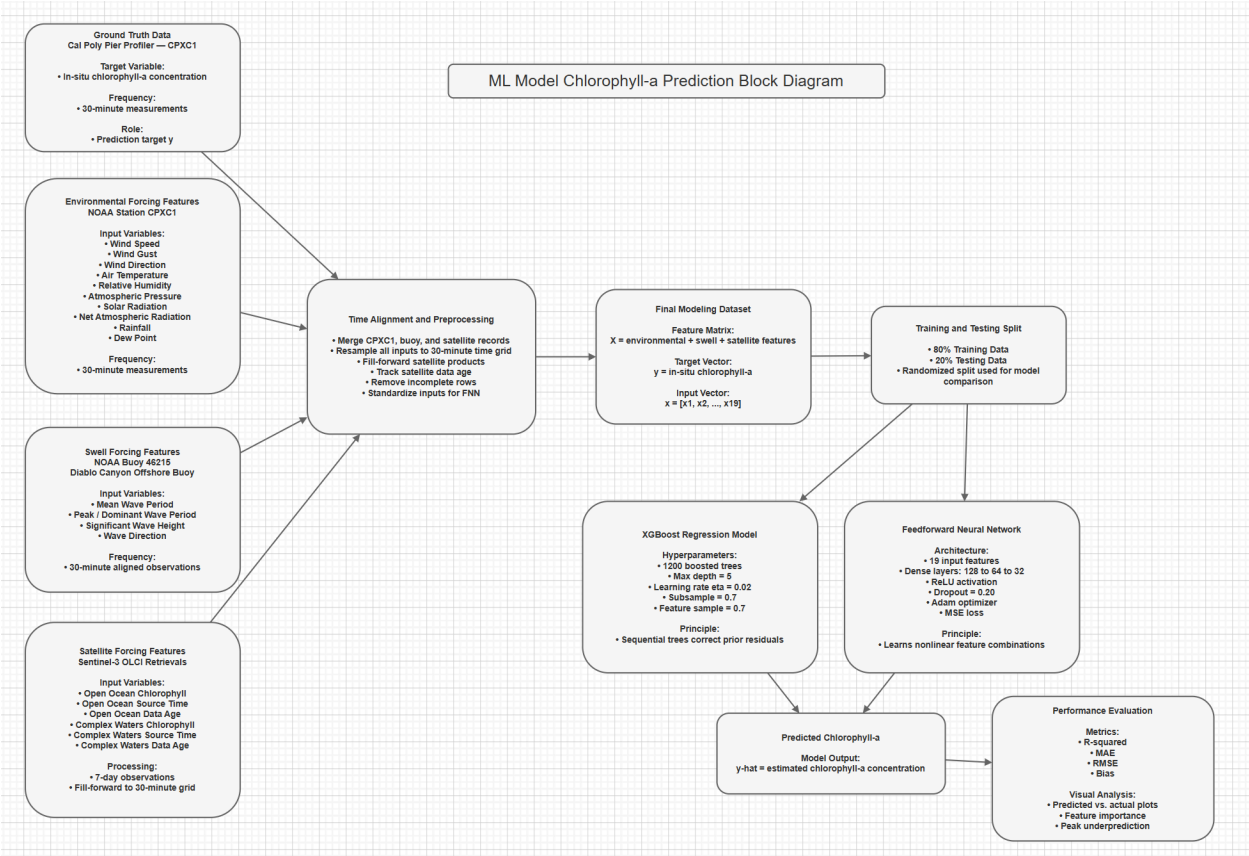

Figure 5.1: High level block diagram

5.1 Study Area and Data Sources

This research aims to analyze how environmental variables relate to chlorophyll values and determine if the inclusion of these variables in machine learning algorithms benefits chlorophyll concentration predictions. The study was conducted using data from the Cal Poly pier in Port San Luis California at NOAA station CPXC1. This location includes in-situ chlorophyll measurements from January 2020 to December 2024 in 30-minute increments from the Cal POly pier profiler. The breadth of this data allowed for a machine learning models to be implemented.

Four sources of data are used in this research. Environmental sources are from the NOAA station CPXC1 (the Cal Poly pier) and include wind speed, wind gust, wind direction, air temperature, relative humidity, atmospheric pressure, solar radiation, net atmospheric radiation, rainfall, and dew point. Swell data is from the closest swell buoy to the Cal Poly pier which is at NOAA buoy station 46215 located offshore of Diablo Canyon. This swell buoy provides mean wave period, dominant wave period, significant wave height, and wave direction. The final input features are from the Sentinel-3 Satellite-derived chlorophyll estimates from Ocean Land and Colour Instruments (OCLI). The in-situ (ground truth) measurements are collected from NOAA station CPXC1 (the Cal Poly pier). It is important to note that all the data except the satellite data is time aligned in 30 minute intervals for the time span 2020 to 2025. The Satellite values are only available and valid every 1 to 7 days and. In order to align these with the other input features and ground truth values the satellite data is carried forward from the last valid value every 30 minutes until the next valid satellite retrieval value.

5.2 Motivation for Environmental and Swell Forcing Features

Historically remote chlorophyll retrieval methods have relied on satellites to retrieve the optical properties of water. These methods stemmed from the OCx algorithms that O'reilly et al. derived which estimate chlorophyll concentrations from the ratios of blue and green wavelengths reflected back to the satellite. These approaches tend to work well in open ocean conditions where ocean conditions are relatively homogeneous over the space and time for which they are measured. However, Satellite estimates perform poorly in coastal environments where external factors such as swell, and environmental features rapidly change chlorophyll levels.

The analysis of 389 time-aligned Sentinel-3 chlorophyll values compared to in-situ retrievals indicates a substantial difference between estimated and measured chlorophyll concentration at the Cal Poly pier. Both Sentinel-3 open ocean models and Sentinel-3 complex waters models were evaluated for the accuracy of chlorophyll predictions. Both models significantly underestimated chlorophyll peaks and had small correlation coefficients, indicating a lack of predictive power in estimating chlorophyll concentrations at the Cal Poly pier. The linear regression indicates that satellite-based chlorophyll predictions are a weak predictor of ground truth chlorophyll concentrations. While a positive relationship exists between the satellite estimates and in-situ chlorophyll values for both the complex waters and open ocean data, the slope of the regression line is 0.071 for the open ocean data, and .117 for the complex waters, which are very small, indicating that a large spike in satellite chlorophyll predictions only correlates to a small increase in measured values at the Cal Poly pier. Additionally, there is a low coefficient of determination $R^2 = 0.201$ for the open ocean estimates and R^2=.112 for the complex waters estimates. The widespread trendline also highlights the frequency of the satellite predictions overestimating the chlorophyll concentrations. This analysis of Sentinel-3 Open Ocean and Complex Waters products supports the hypothesis that satellite products perform poorly in open ocean conditions.

The Sentinel-3 satellite predictions are derived via Copernicus algorithms which measure water leaving reflectance at various visible wavelengths and apply either a OC4Me band-ratio algorithm for the open ocean model or a neural-network algorithm (CHL_NN/C2RCC) for the optically complex waters. These algorithms then convert the reflectance values to chlorophyll estimates. These algorithms are iterations of those presented in table 4.1 [7] [8].

The Sentinel-3 Open Ocean and Complex Optical Coastal waters estimates data used in figures 5.2-5.5 were retrieved through Copernicus Data Space Ecosystem. The data was extracted specifically for the Cal Poly pier (35.170° N, 120.741° W) so that it could be compared for accuracy against the in-situ chlorophyll measurements from the Cal Poly pier profiler. Both the OC4Me for case-1 open ocean conditions and the CHL_NN/C2RCC for case-2 optically complex waters were analyzed. In open ocean conditions phytoplankton are the dominant optical property of water. In optically complex coastal environments, suspended sediments from runoff and upwelling are the optically dominant component of water. The Cal poly pier is located about half a mile off shore and thus both the open ocean product and complex waters product were analyzed for accuracy of chlorophyll predictions at the Cal Poly Pier [8].

The gap between satellite estimates and in-situ values at the Cal Poly pier supports a need to integrate local conditions, such as swell and environmental variables, into the predictions of chlorophyll concentrations. Machine learning models can be used to learn the relationships between these local environmental variables that are not encapsulated by the temporal and spatial gaps in the satellite data. This overarching hypothesis in this research is that environmental forcing features correlate to phytoplankton growth and transport mechanisms in a way that satellites cannot fully capture.

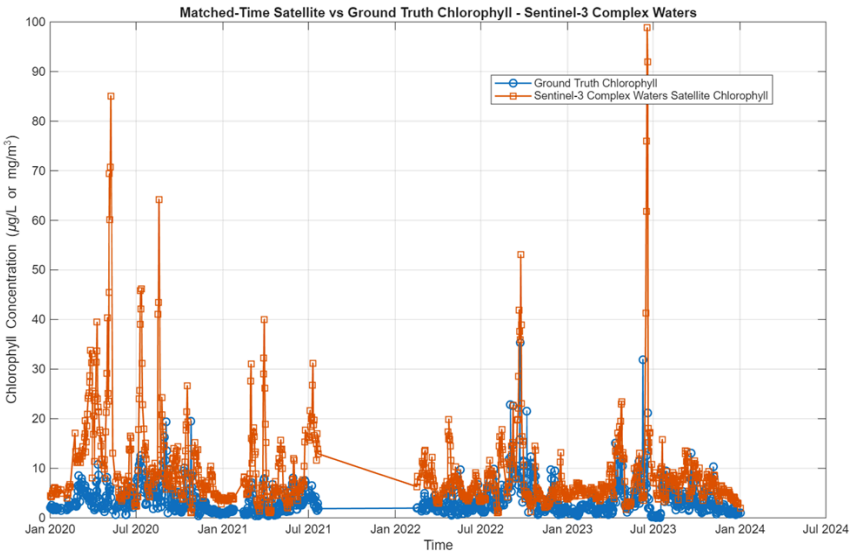

Figure 5.2: Comparison of Time-aligned Chlorophyll Ground Truth and Sentinel-3 Complex Waters Model Chlorophyll Predictions

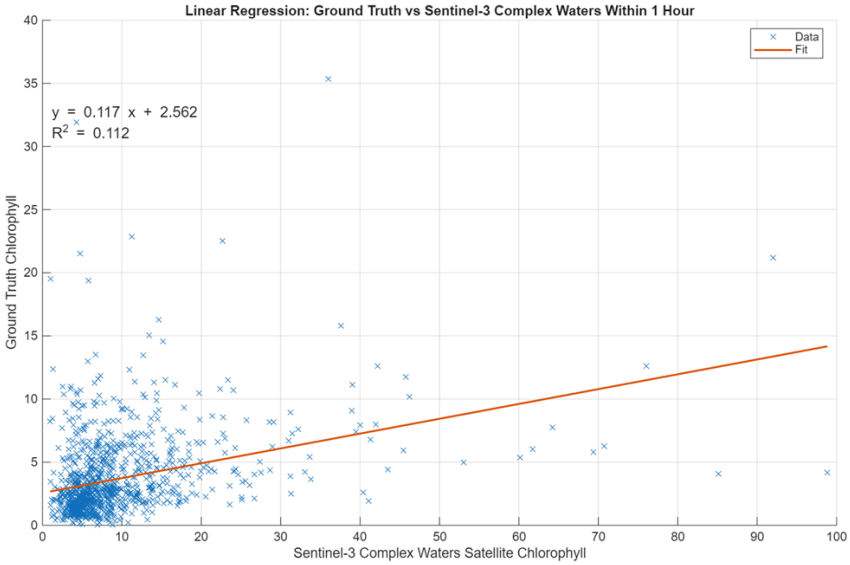

Figure 5.3: Linear regression between the time-aligned Ground Truth Chlorophyll values and Sentinel 3 satellite chlorophyll complex waters model predictions

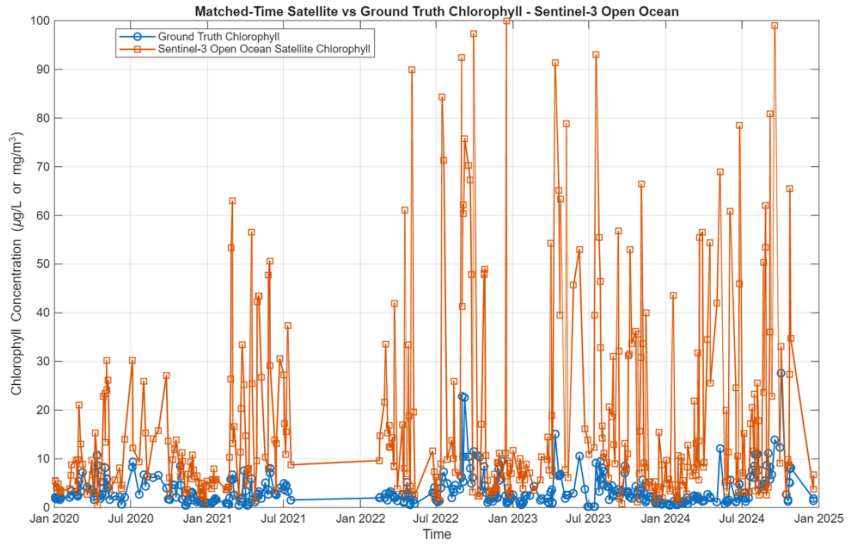

Figure 5.4: Comparison of Time-aligned Chlorophyll Ground Truth and Sentinel-3 Open Ocean Model Chlorophyll Predictions

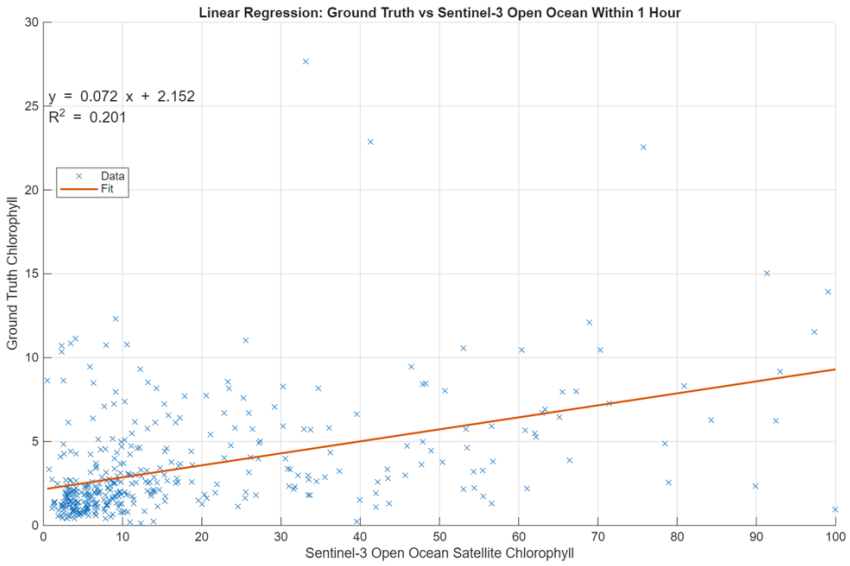

Figure 5.5: Linear regression between the time-aligned Ground Truth Chlorophyll values and Sentinel 3 satellite chlorophyll open ocean Model predictions

5.3 Data Alignment and Missing Data Processing

One challenge of this research was integrating and time-aligning multiple datasets. The Sentinel-3 open ocean and complex waters OLCI data were retrieved from the Copernicus Data Space Ecosystem. These datapoints were at a substantially lower frequency than the environmental, swell, and ground truth values. The data was only available every 1-7 days, with 389 data points total for the 5-year time period [8]. Because of this, the satellite values were regrouped into 30-minute buckets to be time-aligned with the environmental swell and ground truth data. Values were filled forward until the next valid satellite datapoint. Age of the value is also stored so that the models give more weight to more recent satellite data.

In-situ chlorophyll measurements and environmental variables were taken from the Cal Poly Pier station CPXC1 through the AXDS ERDDAP data portal from 2020 to 2024 over 30-minute intervals. Offshore swell conditions were taken from NOAA Station 46215 off of Diablo Canyon, also from the ERDDAP data portal [9] [10]. A standardized dataset was then created with all data types (in-situ, environmental, swell, satellite), time aligned in 30-minute increments for the 5-year time period. Figure 5.6 displays the gap in each datatype. Each time, a gap in any of the data was removed from the testing and training dataset. This was done within the Machine learning algorithms.

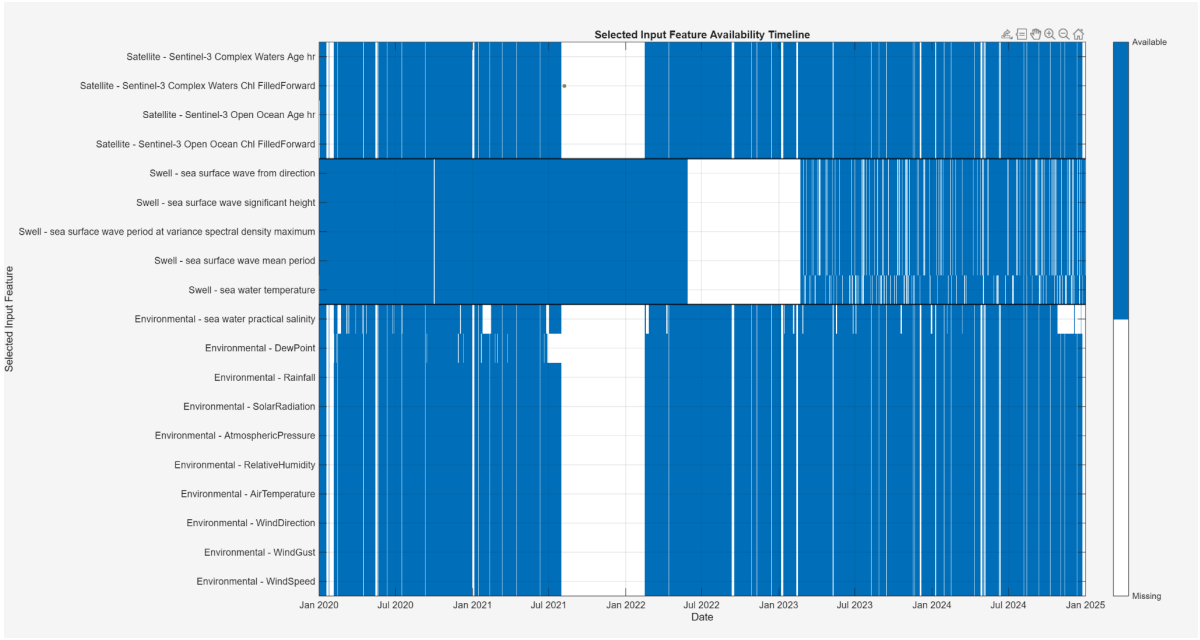

Figure 5.6: Data Availability- Environmental, Swell, and Filled Forward Satellite

5.4 Feature Selection and Input Vector Construction

Ultimately, the input vector to the machine learning algorithms consisted of environmental, swell, and satellite forcing features. The environmental features are shown in Table 5.1, the swell forcing features in Table 5.2, and the Satellite forcing features in Table 5.3.

Each environmental and swell variable was selected because of its suspected correlation to chlorophyll suspension and growth. Satellite chlorophyll estimates provide an optical observation of phytoplankton concentrations. The goal is that environmental variables will help fill the temporal and spatial gaps in satellite observations.

The FNN and XGBoost machine learning models were trained to estimate concentrations of chlorophyll in micrograms per liter using regressive models. The machine learning models were trained to estimate chlorophyll-a concentration, measured in micrograms per liter, using these forcing variables as predictors.

Variable	Description	Units
Wind Speed	Average wind speed	m/s
Wind Gust	Maximum wind speed	m/s
Wind Direction	Wind direction	degrees
Air Temperature	Atmospheric temperature	°C
Relative Humidity	Atmospheric moisture	%
Atmospheric Pressure	Surface pressure	hPa
Solar Radiation	Incoming solar energy	W/m²
Net Atmospheric Radiation	Net radiative exchange	W/m²
Rainfall	Precipitation accumulation	mm
Dew Point	Moisture condensation temperature	°C

Table 5.1: Environmental Forcing Variables

Variable	Description	Units
Mean Wave Period	Average interval between successive waves	s
Peak (Dominant) Wave Period	Wave period matched to the maximum energy in the wave spectrum	s
Significant Wave Height	Average height of the highest one-third of observed waves	m
Wave Direction	The direction in which the waves are propagating	degrees

Table 5.2: Swell Forcing Variables

Variable	Description	Units
Open Ocean Chlorophyll (Filled Forward)	Sentinel-3 open ocean chlorophyll-a estimate (OC4Me algorithm). Missing satellite observations are propagated forward.	µg/L
Open Ocean Source Time	Timestamp of the original Sentinel-3 open ocean chlorophyll observation used to populate the filled-forward value.	UTC datetime
Open Ocean Data Age	Time elapsed between the model timestamp and the original open ocean satellite observation. Used to indicate how stale the filled-forward value is.	hr
Complex Waters Chlorophyll (Filled Forward)	Sentinel-3 optically complex waters chlorophyll-a estimate (C2RCC algorithm). Missing observations are filled forward using the most recent valid retrieval.	µg/L
Complex Waters Source Time	Timestamp of the original Sentinel-3 complex waters chlorophyll observation used to populate the filled-forward value.	UTC datetime
Complex Waters Data Age	Time elapsed between the model timestamp and the original complex waters satellite observation. Indicates the confidence in the temporal alignment of the satellite estimate.	hr

Table 5.3: Satellite Forcing Variables

5.5 XGBoost Regression Model

In this study, the eXtreme Gradient Boosting (XGBoost) model was designed, trained, and tested first. XGBoost relies on sequential trees to regressively correct the error of the previous tree. The final prediction is a sum of the contributions of each tree. XGBoost trees are constructed using a branching binary decision split, where at every node, the feature and the threshold are determined by the algorithm. If a sample makes the tree split step of table 5.x true, then the sample moves left through the tree, and if the condition is false, then the sample moves right through the tree.

The XGBoost model used in this research uses 1200 decision trees and was optimized for adequate fitting of the data without overfitting and memorizing noise. Additionally, a learning rate of 0.02 was used to reduce overfitting and allow for gradual optimization. Row sampling ratios of .7 were applied so that for each tree not all rows were used. This allows the trees to learn distinct patterns and reduce overfitting. The parameters were experimented with and modified substantially to find the highest correlation while minimizing overfitting. The XGBoost model is beneficial because it can learn non-linear relationships, which are prevalent in this chlorophyll data.

The XGBoost model used in this study contained 1200 decision trees with a maximum tree depth of five. A learning rate of 0.02 was selected to ensure gradual optimization and reduce overfitting. Row and

feature subsampling ratios of 0.7 were applied during training. These parameters were selected through experimentation and are consistent with common practices for environmental regression problems. The model is advantageous because it naturally handles nonlinear relationships, variable interactions, and mixed feature types without requiring extensive feature engineering. Figure 5.x shows the structure of the XGBoost algorithm in terms of input features, decision trees, and outputs.

The core XGBoost model parameters set in the code are defined in Figure 5.7 (see Appendix A). Additionally, these parameters are listed in a table along with the equations that define the structure of the XGBoost algorithm.

```
# Train model using regressive methods
model = XGBRegressor(
    n_estimators=1200,
    max_depth=5,
    subsample=0.7,
    colsample_bytree=0.7,
    reg_alpha=0.5,
    reg_lambda=2.0,
    min_child_weight=4,
    objective="reg:squarederror",
    n_jobs=-1,
    random_state=42
)
```

Figure 5.7: Core XGBoost Model Structure

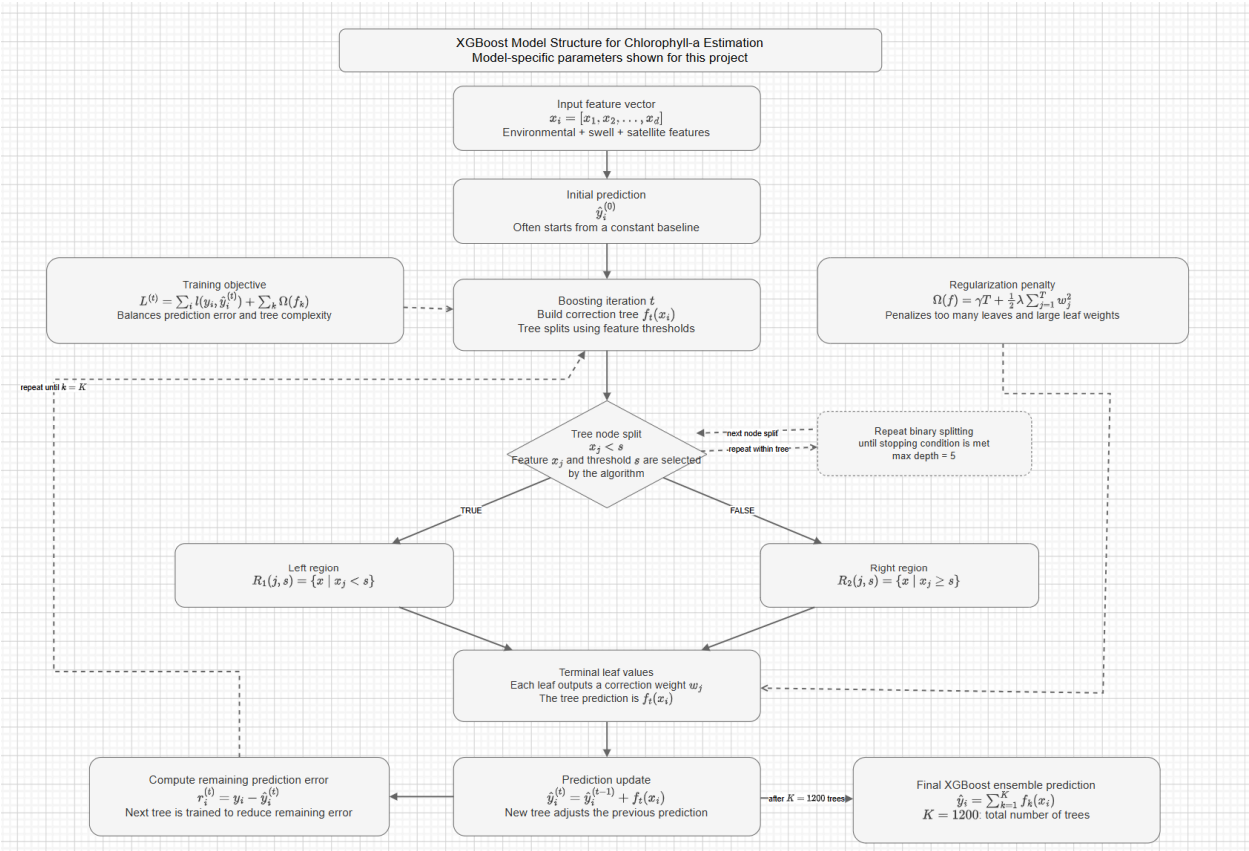

Figure 5.10: XGBoost Block diagram

Step	Value	Function	Equation
Input layer	19 inputs	Environmental + swell + satellite inputs	$x = [x_1, x_2, \dots, x_{19}]$
Tree ensemble	1200 trees	Sequentially improve prediction	$F(x) = \sum_{i=1}^{1200} \eta\, T_i(x)$
Single tree depth	max depth = 5	Maximum of 5 decision levels per tree	Leaves $\leq 2^5 = 32$
Tree split	Binary decision	Split data into environmental regions	$x_j < threshold$
Residual update	Error correction	Next tree predicts remaining error	$r = y - \hat{y}$
Learning rate	$\eta = 0.02$	Controls contribution of each tree	$\hat{y}_k = \hat{y}_{k-1} + \eta T_k(x)$
Subsample	0.7	Uses 70% of rows per tree	Random training subset
Feature sample	0.7	Uses 70% of inputs per tree	Random feature subset
Final output	1 prediction	Combined chlorophyll estimate	$\hat{y} = F(x)$

Table 5.4: XGBoost Model Architecture

5.6 Feedforward Neural Network Model

The other machine learning algorithm investigated was the Feedforward Neural Network. Neural networks are connected layers of computational nodes that learn non-linear patterns in datasets. The input features in tables 5.1-5.3 are then propagated through each 3 of the hidden layers. There are 128 neurons in layer 1, 64in layer 2, and 32 in layer 3. Each neuron then computes the weighted combination of inputs using the Rectified Linear Unit (ReLU) activation. This activation function introduces a non-linear activation.

In the first hidden layer, primary feature extraction occurs. Here, the environmental, swell, and satellite inputs are transformed into 128 learned feature relationships. These represent prevalent combinations of input variables that indicate changes in chlorophyll concentrations in a particular direction. The second layer of 64 neurons relates non-linear interactions between groups of inputs rather than individual inputs. Finally, the third layer condenses its inputs into the decision layer of learned feature combinations. The final output layer is made up of one neuron, which produces the chlorophyll-a prediction. The equations used in each step of this method are shown in Table 5.x.

The predicted chlorophyll value is compared to the in-situ ground truth values during training based on the mean squared error (MSE). Then the prediction error is backpropagated through the neural network

using the backpropagation gradients so that the gradient of loss can be computed for each network weight. The weight contributions are then updated based on the gradients of loss which indicate the contributions of prediction error from each network weight. To iteratively adjust these weights the Adam optimization algorithm was used.

The core FNN model parameters set in the code are defined in figure 5.9 (see appendix B). Additionally these parameters are listed in table along with the equations which define the structure of the FNN algorithm.

```
# FNN model architecture
#This architecture was adapted for my project from keras.io developer guides[1]
model = Sequential([
    #layer 1
    Dense(128, activation="relu", input_shape=(X_train_scaled.shape[1],)),
    Dropout(0.20),
    #layer 2
    Dense(64, activation="relu"),
    Dropout(0.20),
    #layer 3
    Dense(32, activation="relu"),
    Dense(1)
])


#Early stop
early_stop = EarlyStopping(
    monitor="val_loss",
    patience=25,#number of epochs without improvement before stopping
    restore_best_weights=True #return to best epoch weight
)

#Train
history = model.fit(
    X_train_scaled,
    y_train,
    validation_split=0.2,#reserve 20% training data for validation
    epochs=300, #number of complete passes through training data allowed
    batch_size=64,
    callbacks=[early_stop],
    verbose=1
)
```

Figure 5.9: Core FNN model structure

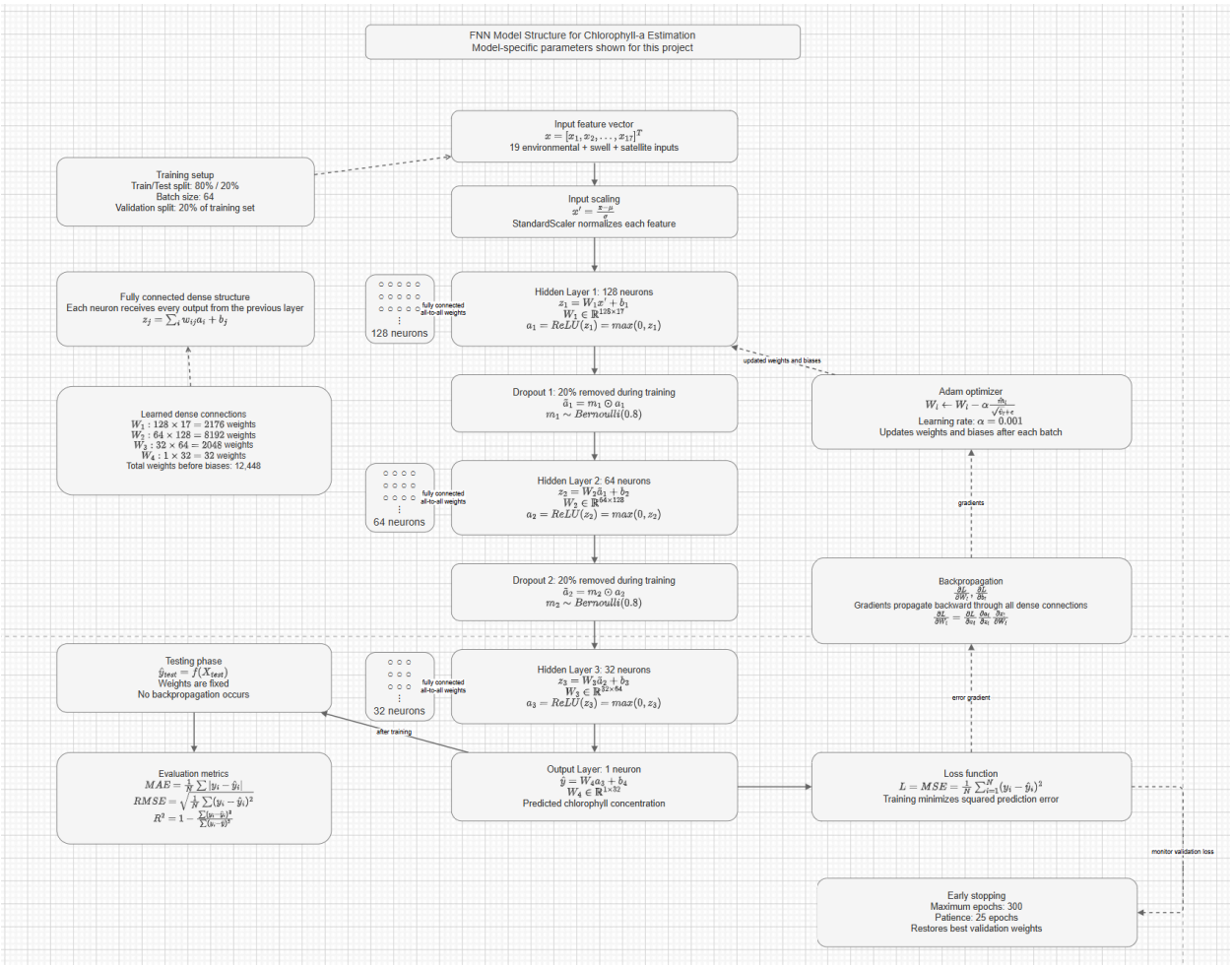

Figure 5.10: FNN Block diagram

Steps	Value	Function	Equation
Input layer	19 inputs	env + swell + satellite inputs	$x = [x_1, x_2, \dots, x_{19}]$
Hidden layer 1	128 neurons	First learned feature-combination layer	$z_1 = W_1 x + b_1$ ReLU: $a_1 = \max(0, z_1)$
Dropout 1	0.20	Randomly removes 20% of neurons	$a_{1,\text{drop}} = m \odot a_1$ where $m \sim Bernoulli(0.8)$
Hidden layer 2	64 neurons	Combines outputs from layer 1	$z_2 = W_2 a_1 + b_2$ ReLU: $a_2 = \max(0, z_2)$
Dropout 2	0.20	Removes 20% of active neurons	$a_{2,\text{drop}} = m \odot a_2$
Hidden layer 3	32 neurons	Final learned feature-combination layer	$z_3 = W_3 a_2 + b_3$ ReLU: $a_3 = \max(0, z_3)$
Output layer	1 neuron	Produces chlorophyll prediction	$\hat{y} = W_4 a_3 + b_4$

Table 5.5: FNN Model Architecture

5.7 Training and Evaluation Procedure

To allow for fair comparison of both models, each model was trained using identical datasets and evaluation procedures. The dataset used in both the testing and training algorithms was split into 80% training samples and 20% testing samples. The training subset was then used to fit the model parameters and the testing data was used only for performance evaluation. The coefficient of determination, root mean square error,prediction bias and mean absolute error are used to analyze the models.

6. Computer Simulations and/or Experimental Results

6.1 XGBoost Performance Metrics

The XGBoost model obtained stronger performance metrics compared to the FNN model. The model obtained a test R^2 value of 0.853 indicating that over the 5 years trained upon about 85.3% of the variability in chlorophyll can be explained by the environmental, swell, and satellite data. The model also obtained a MAE of 0.794 µg/L and RMSE of 1.483 µg/L which indicate that the model predicted both small and larger fluctuations in chlorophyll levels while slightly underpredicting spikes in chlorophyll.

The training R^2 value of 0.992 shows a moderate test-training gap of 0.139 which suggests moderate but not unreasonable overfitting in the model.

It is important to note that these results are obtained from a randomized 80/20 testing and training split. Accordingly the results represent the models ability to predict chlorophyll values from the time period and seasons in the 5 years that were not used for training and does not necessarily extend into forecasting future years.

Category	Metric	Value	Meaning
Prediction Accuracy	MAE	**0.794**	Average absolute prediction error
Prediction Accuracy	RMSE	**1.483**	Typical prediction error (model is slightly underpredicting)
Model Fit	R^2	**0.853**	Percent of chlorophyll variability explained on test data
Bias	Bias	**0.002**	Average overprediction or underprediction
Generalization	Train R^2	**0.992**	Model fit on training data
Generalization	Test R^2	**0.853**	Model fit on unseen test data
Overfitting	R^2 Gap	**0.139**	Difference between training and testing performance

Table 6.1: XGBoost performance metrics

6.2 XGBoost Predicted vs Actual Chlorophyll Concentration

Figure 6.1-6.3 show the comparison of measured chlorophyll values with XGBoost predictions during the 2020 to 2024 study timeframe. The graph shows that the 20% testing reproduced a trend which follows the seasonal and short term trends of chlorophyll production. While there were slight underpredictions in some of the algal bloom spikes, the model effectively captures the non-linear relationships between environmental, swell, and satellite variables as they relate to the measured chlorophyll concentrations.

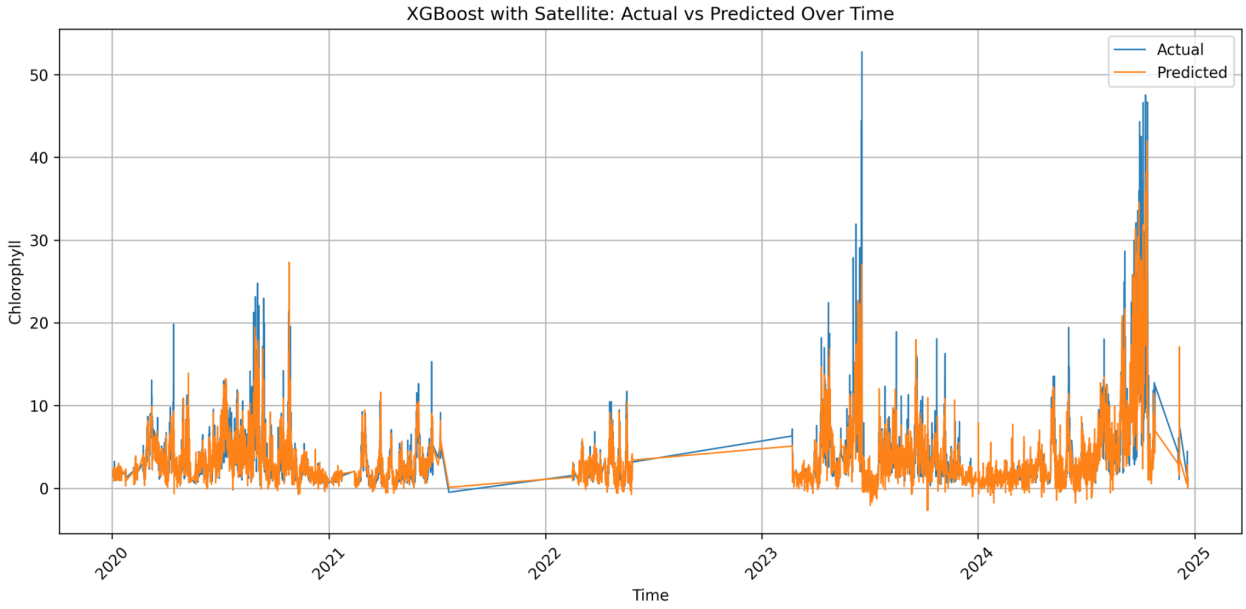

Figure 6.1: XGBoost Predicted vs Actual Chlorophyll Concentrations

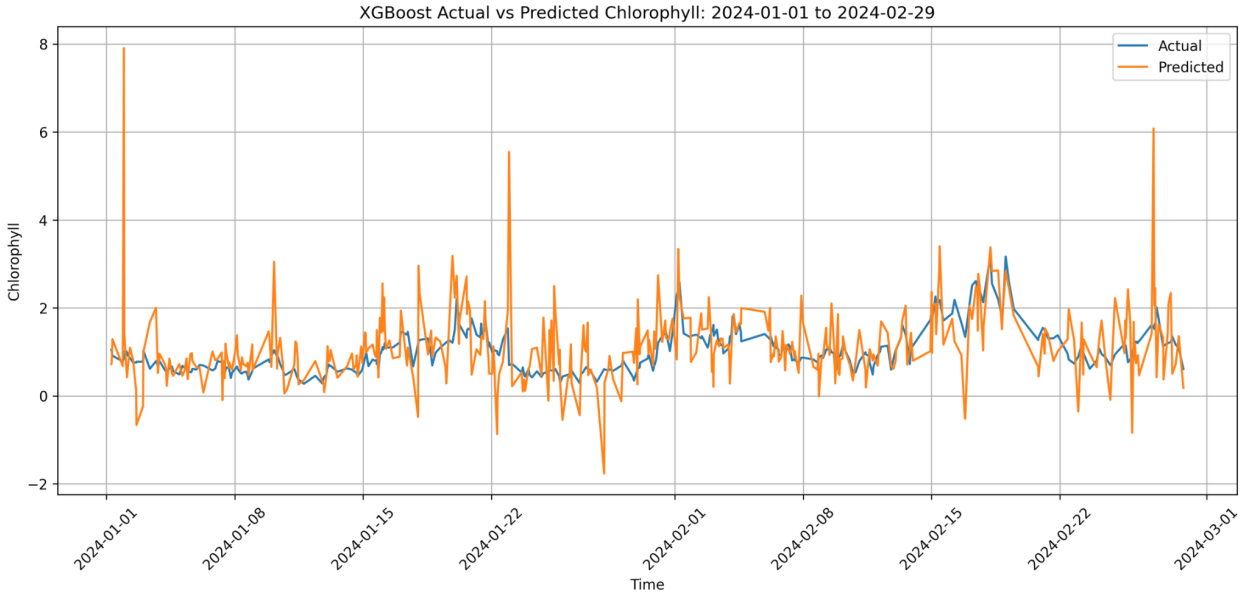

Figure 6.2: XGBoost Predicted vs Actual Chlorophyll Concentrations (Zoomed In: 2024 Jan./Feb.)

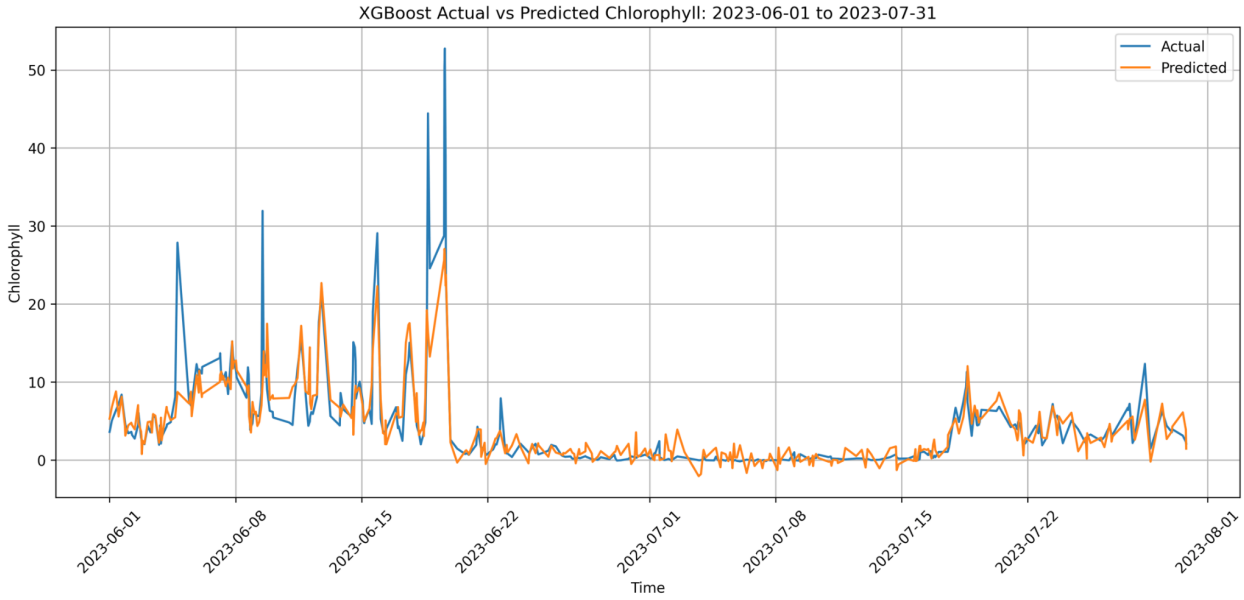

Figure 6.3: XGBoost Predicted vs Actual Chlorophyll Concentrations (Zoomed In: 2023 June/July)

6.3 FNN Performance Metrics

The FNN obtained a decent test R^2 value of 0.745 and an MAE of 1.154 µg/L. While the XGBoost was more successful at matching the trend of the five-year chlorophyll pattern the XGBoost model still performed well in following this trend. Additionally, the small training and testing R^2 gap indicates that there was limited overfitting, meaning the dropout layers and patience values were effective in moderating overfitting. The main difference between the performance of the XGBOost and FNN is that the FNN did not capture the bloom spikes as well. This can be seen through the larger RMSE for the FNN model.

Category	Metric	Value	Interpretation
Prediction Error	MAE	1.1544	Average prediction error ≈ 1.15 µg/L chlorophyll
Prediction Error	RMSE	1.9504	Prediction error with larger errors weighted more heavily
Model Fit	R^2	0.7454	The model explains 74.54% of chlorophyll variability
Bias	Bias	−0.1401	Small tendency to underpredict chlorophyll
Training Performance	Train R^2	0.8146	Model fit on training data(80%)
Testing Performance	Test R^2	0.7454	Predictive performance on unseen data (20%)
Overfitting Check	R^2 Gap	0.0692	A small train–test gap means there is a strong generalization

Table 6.2: FNN performance metrics

6.4 FNN Predicted vs Actual Chlorophyll Concentration

Figure 6.4 -6.6 compare the predicted and measured chlorophyll concentrations over the 5 year time period for the FNN model. The model adequately followed long-term trends and standard chlorophyll fluctuations however it underpredicted significant fluctuation in chlorophyll. This underprediction of spikes is typical for neural network regression models which tend to prioritize trend following over capturing spikes.

measured and predicted chlorophyll concentrations produced by the Feedforward Neural Network. The model accurately reproduced long-term trends and moderate chlorophyll variations. However, several high-magnitude bloom events were either underpredicted or smoothed relative to the observations. This behavior is typical of neural-network regression models trained using mean squared error, which tend to prioritize overall predictive accuracy over precise reproduction of extreme events[6]. This occurs because the MSE corrects weights throughout the model and trends towards chlorophyll averages which does not account well for the spikes.

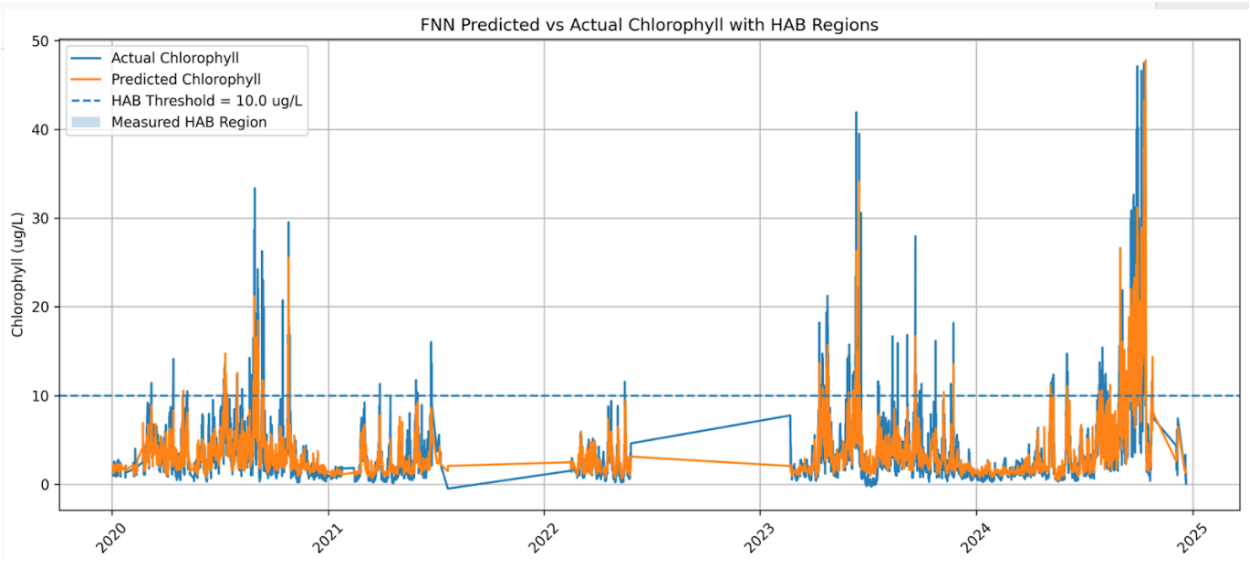

Figure 6.4: FNN Predicted vs Actual Chlorophyll Concentrations (2020-2024)

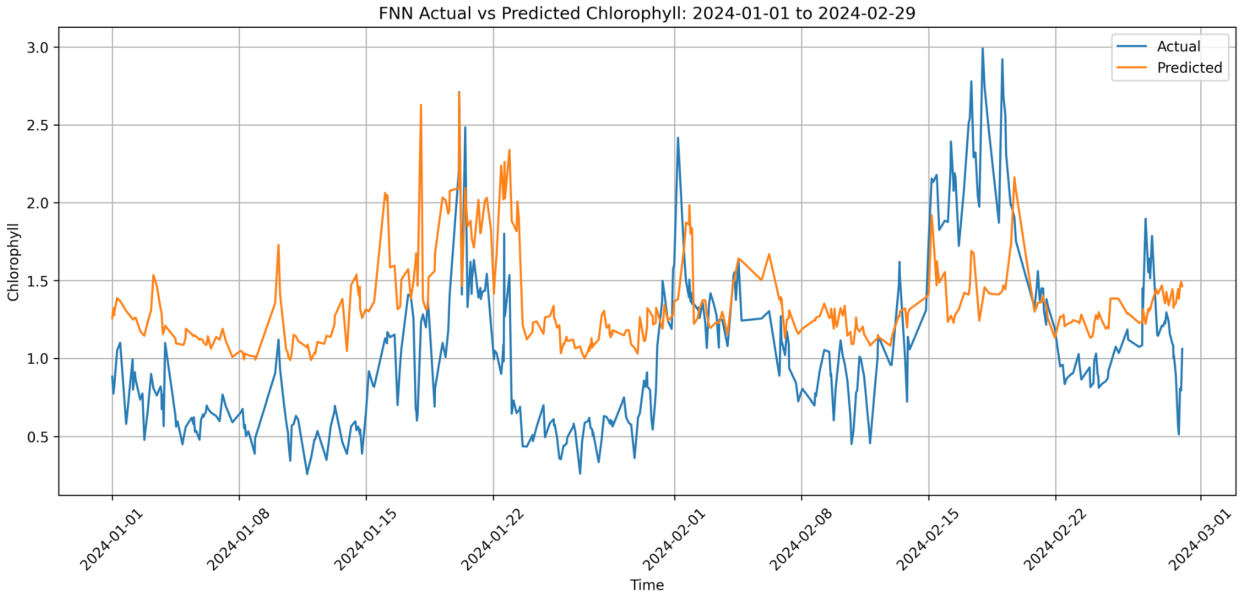

Figure 6.5: FNN Predicted vs Actual Chlorophyll Concentrations (Zoomed In: 2024 Jan./Feb.)

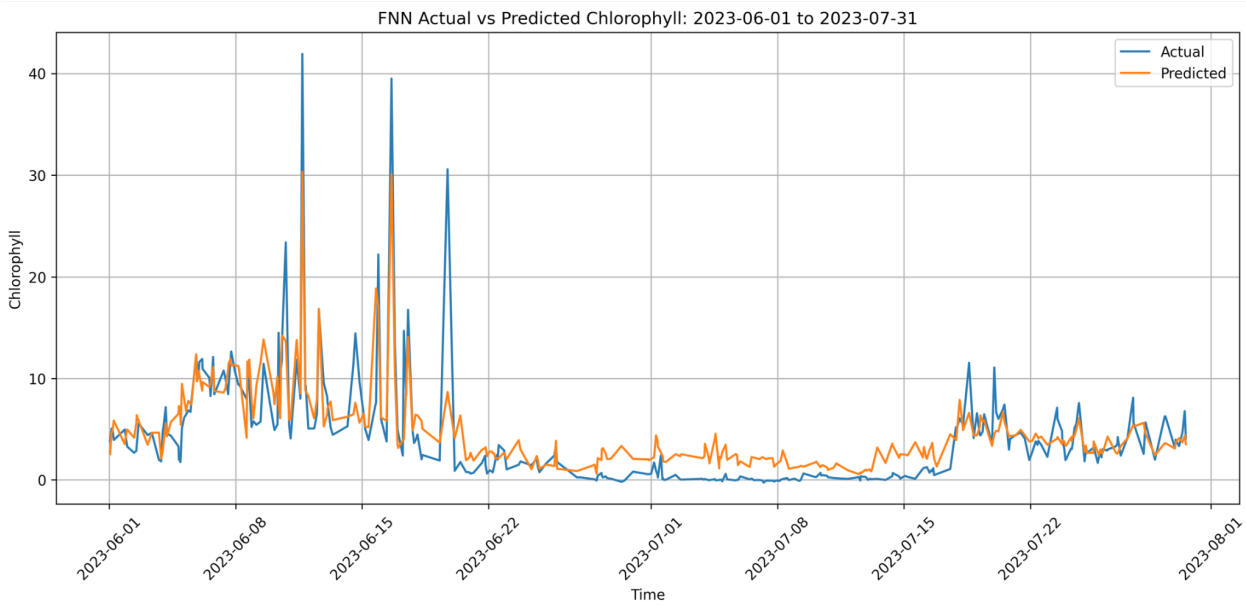

Figure 6.6: FNN Predicted vs Actual Chlorophyll Concentrations (Zoomed In: 2023 June/July)

7. Conclusions and Future Works

This research compares the effectiveness of XGBoost and FNN models in estimating chlorophyll-a at the Cal Poly Pier over a 5-year period. The model relies on time-aligned environmental, swell, satellite, and in-situ measurements from 2020 to 2024 taken every 30 minutes, with the satellite data relying on filled forward values.

The results indicate that including environmental and swell features improves the quality of chlorophyll predictions compared to solely using satellite observations. Analysis if the Sentinel-3 satellite predictions for the Cal Poly Pier displays weak correlation between satellite estimates for both the open ocean and complex waters models in predicting chlorophyll concentrations at the Cal Poly pier. These results support the hypothesis that local coastal processes, both swell driven and environmental, affect chlorophyll concentrations in ways that cannot be fully captured by satellite.

Both the XGBoost and FNN models successfully captured the long-term trend of the chlorophyll bloom event. The XGBoost model, however, was more successful at modeling spikes in chlorophyll concentrations and presented stronger predictive power overall. These results suggest that tree-based models may be better at learning and modeling the highly non-linear relationships between environmental features than the generalized function approximations learned using the FNN model.

A limitation of this study was the randomized 80/20 train-test split that was used. While this is commonly used in environmental models, it demonstrates the ability of the model to predict unseen chlorophyll from the timeframe that was tested on, rather than predicting an unseen future. Future work should include a year holdout validation where the model is trained on a few years and tested on an unseen year. Additional research could also focus on analyzing how transferable the model for the Cal Poly pier is to other coastal monitoring.

Ultimately, this research indicates that integrating local environmental and swell measurements with satellite observations significantly improves the chlorophyll estimates in optically complex waters compared to using satellites alone.

8. List of References

Appendix: Put long derivations and/or computer program (code) here

IEEE Citation:

[1] Q. S. Qing, B. Zhang, and X. Zhang, "Improving remote sensing retrieval of water clarity in complex coastal and inland waters with modified absorption estimation and optical water classification," *International Journal of Applied Earth Observation and Geoinformation*, vol. 102, p. 102323, 2021.

[2] C. Brockmann, R. Doerffer, M. Peters, S. Kerstin, J. Embacher, and A. Ruescas, "Evolution of the C2RCC neural network for Sentinel 2 and 3 for the retrieval of ocean colour products in normal and extreme optically complex waters," in *Proc. ESA Living Planet Symposium*, 2016.

[3] A. Ali, G. Zhou, F. P. A. Lopez, C. Xu, G. Jing, and Y. Tan, "Deep learning for water quality multivariate assessment in inland water across China," *International Journal of Applied Earth Observation and Geoinformation*, vol. 133, p. 104078, 2024.

[4] L. Yao, X. Wang, and J. Zhang, "Prediction of sea surface chlorophyll-a concentrations based on deep learning and time-series remote sensing data," *Remote Sensing*, vol. 15, no. 18, p. 4486, 2023.

[5] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, New York, NY, USA, 2016, pp. 785–794.

[6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[7] J. E. O'Reilly et al., "Ocean chlorophyll algorithms for SeaWiFS," *Journal of Geophysical Research: Oceans*, vol. 103, no. C11, pp. 24,937–24,953, 1998.

[8] European Space Agency and Copernicus Data Space Ecosystem, *Sentinel-3 OLCI Level-2 Water Products*. Accessed: Jun. 5, 2026.

[9] AXDS, "Cal Poly Pier Shore Station (CPXC1) Environmental Monitoring Data," AXDS ERDDAP Data Server. Accessed: Jun. 5, 2026.

[10] National Data Buoy Center (NDBC), "Station 46215 Historical Data, Diablo Canyon Offshore, California," National Oceanic and Atmospheric Administration (NOAA). Accessed: Jun. 5, 2026.

9. Appendix

A. XGBoost Code

```
# XGBoost Chlorophyll Model with Environmental + Swell + Satellite Inputs

import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from xgboost import XGBRegressor, plot_tree
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score


# File path

base_path = r"C:\Users\jorda\Documents\Masters
program\Thesis\Raw_data_tables\Finalized_data"

input_file = "MASTER_with_satellite_and_clean_swell_30min.csv"
file_path = os.path.join(base_path, input_file)

output_folder = os.path.join(base_path, "XGBoost_swell_satellite_outputs")
os.makedirs(output_folder, exist_ok=True)

print("check if input folder exists", os.path.exists(file_path))
print("the output folder:", output_folder)


# Load data

df = pd.read_csv(file_path)

print("\nOriginal rows:")
print(len(df))

print("\nColumns:")
print(df.columns.tolist())
```

```
#Features and target (ground truth values)

features = [
    # Environmental variables
    "WindSpeed",
    "WindGust",
    "WindDirection",
    "AirTemperature",
    "RelativeHumidity",
    "AtmosphericPressure",
    "SolarRadiation",
    "Rainfall",
    "DewPoint",
    "sea_water_practical_salinity",

    # Swell variables
    "sea_water_temperature_SWELL",
    "sea_surface_wave_mean_period_SWELL",
    "sea_surface_wave_period_at_variance_spectral_density_maximum_SWELL",
    "sea_surface_wave_significant_height_SWELL",
    "sea_surface_wave_from_direction_SWELL",

    # Satellite baseline chlorophyll variables
    "Sentinel3_OpenOcean_Chl_FilledForward",
    "Sentinel3_OpenOcean_Age_hr",
    "Sentinel3_ComplexWaters_Chl_FilledForward",
    "Sentinel3_ComplexWaters_Age_hr"
]

target = "mass_concentration_of_chlorophyll_in_sea_water"
time_col = "TimeUTC_30min"

required_cols = features + [target, time_col]

#store the missing columns (built with CODEX suggestions)
missing_cols = [col for col in required_cols if col not in df.columns]

if missing_cols:
    raise ValueError(f"there are required columns missing: {missing_cols}")


# Data cleaning
```

```
#keep only needed columns for model
model_df = df[required_cols].copy()

#convert time column to timestamps (used CODEX suggestions)
model_df[time_col] = pd.to_datetime(
   model_df[time_col],
   errors="coerce"
)

#used to convert each feature and ground truth to values
for col in features + [target]:
   model_df[col] = pd.to_numeric(
      model_df[col],
      errors="coerce"
   )

#totalrows before cleaning
rows_before = len(model_df)

# save which rows were missing values
removed_rows = model_df[
   model_df[features + [target]].isna().any(axis=1)
].copy()

#remove any datapoint for all columns that is missing in any of the inputs
#or the ground truths
model_df = model_df.dropna(
   subset=features + [target]
)

rows_after = len(model_df)

print("\nRows the model was cleaned:")
print(rows_before)

print("\nRows that were removed because of missing features or ground truth value:")
print(rows_before - rows_after)

print("\nRows used for the model:")
print(rows_after)

removed_rows_file = os.path.join(
   output_folder,
```

```
    "XGB_removed_rows_missing_inputs.csv"
)

removed_rows.to_csv(
    removed_rows_file,
    index=False
)


#Training/testing This code is based on the XGBoost Scikit-Learn implimentation
#examples and addapted for my dataset and optimization of my parameters [1]

#Create an 80/20 training and testing split
X = model_df[features] #env., swell, and sattelite data
y = model_df[target] #chlorophyll ground truth data

X_train, X_test, y_train, y_test, time_train, time_test = train_test_split(
    X,
    y,
    model_df[time_col],
    test_size=0.2,
    random_state=31
)

print("\nTraining rows:")
print(len(X_train))

print("\nTesting rows:")
print(len(X_test))


#Train model

model = XGBRegressor(
    n_estimators=1200, #of trees
    max_depth=5, #max depth of branches on the tree
    subsample=0.7,#fraction of data rows per tree
    colsample_bytree=0.7, #fraction of features per tree
    reg_alpha=0.5, #more weight to simpler trees
    reg_lambda=2.0, #less weight for large corrections
    min_child_weight=4, #min rows to split so we dont split small env. groups
    objective="reg:squarederror", #loss function: L = (y - y_pred)^2
    n_jobs=-1, #use all CPU cores
    random_state=31
```

```
)

model.fit(X_train, y_train)


#predict using the trained model (used codex varriable suggestions)
y_pred = model.predict(X_test)
y_train_pred = model.predict(X_train)


# metrics (used codex varriable suggestions)
mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)
bias = np.mean(y_pred - y_test)

train_r2 = r2_score(y_train, y_train_pred)
test_r2 = r2
r2_gap = train_r2 - test_r2

#(used codex formatting suggestions)
print("\nXGBoost results from environmental swell and satellite inputs")
print(f"MAE {mae:.4f}")
print(f"RMSE {rmse:.4f}")
print(f"Test R² {r2:.4f}")
print(f"average bias {bias:.4f}")

print("\nOverfitting check")
print(f"training R²{train_r2:.4f}")
print(f"testing R² {test_r2:.4f}")
print(f" training - testing R² was {r2_gap:.4f}")


#Result summary table

results_summary_df = pd.DataFrame({
   "Metric": [
      "MAE",
      "RMSE",
      "R2",
      "Bias",
      "Train_R2",
      "Test_R2",
      "R2_Gap"
```

```
    ],
    "Value": [
        mae,
        rmse,
        r2,
        bias,
        train_r2,
        test_r2,
        r2_gap
    ]
})

metrics_file = os.path.join(
    output_folder,
    "XGB_results_summary_table.csv"
)

results_summary_df.to_csv(
    metrics_file,
    index=False
)

print("\nResults summary table:")
print(results_summary_df)


#Save predictions

results_df = pd.DataFrame({
    "TimeUTC_30min": time_test.values,
    "Actual_Chlorophyll": y_test.values,
    "Predicted_Chlorophyll": y_pred,
    "Residual": y_pred - y_test.values
})

results_df = results_df.sort_values("TimeUTC_30min")

predictions_file = os.path.join(
    output_folder,
    "XGB_swell_satellite_predictions_vs_actual.csv"
)

results_df.to_csv(
    predictions_file,
```

```
    index=False
)


#Feature importance this tells which features was the most correlated
#to the changes in chlorophyll

# get importance of each feature
importance_df = pd.DataFrame({
    "Feature": features,
    "Importance": model.feature_importances_
})

#order from most to least important
importance_df = importance_df.sort_values(
    "Importance",
    ascending=False
)
#designate the output file
importance_file = os.path.join(
    output_folder,
    "XGB_swell_satellite_feature_importance.csv"
)

#save each importance value
importance_df.to_csv(
    importance_file,
    index=False
)

#Codex suggestions used for plot formatting
plt.figure(figsize=(12, 8))

plt.barh(
    importance_df["Feature"],
    importance_df["Importance"]
)

plt.gca().invert_yaxis()
plt.xlabel("Feature Importance")
plt.ylabel("Feature")
plt.title("XGBoost Feature Importance: Environment, Swell, & Satellite")
plt.grid(True)
```

```
importance_plot_file = os.path.join(
    output_folder,
    "XGB_swell_satellite_feature_importance.png"
)

plt.savefig(
    importance_plot_file,
    dpi=300,
    bbox_inches="tight"
)

plt.close()



#Predicted vs. measured scatter plots
plt.figure(figsize=(8, 8))

plt.scatter(
    y_test,
    y_pred,
    alpha=0.5
)

min_val = min(y_test.min(), y_pred.min())
max_val = max(y_test.max(), y_pred.max())

plt.plot(
    [min_val, max_val],
    [min_val, max_val],
    linestyle="--"
)

plt.xlabel("Actual Chlorophyll")
plt.ylabel("Predicted Chlorophyll")
plt.title("XGBoost with Satellite: Predicted vs Actual Chlorophyll")
plt.grid(True)

scatter_file = os.path.join(
    output_folder,
    "XGB_swell_satellite_predicted_vs_actual.png"
)

plt.savefig(
```

```
    scatter_file,
    dpi=300,
    bbox_inches="tight"
)

plt.close()


#Time series plots
#these show over the 5 year period the difference between
#predicted in orange and in-situ chlorophyl values in blue
plt.figure(figsize=(14, 6))

plt.plot(
    results_df["TimeUTC_30min"],
    results_df["Actual_Chlorophyll"],
    label="Actual",
    linewidth=1
)

plt.plot(
    results_df["TimeUTC_30min"],
    results_df["Predicted_Chlorophyll"],
    label="Predicted",
    linewidth=1
)

plt.xlabel("Time")
plt.ylabel("Chlorophyll")
plt.title("XGBoost with Satellite: Actual vs Predicted Over Time")
plt.legend()
plt.grid(True)
plt.xticks(rotation=45)

timeseries_file = os.path.join(
    output_folder,
    "XGB_swell_satellite_actual_vs_predicted_timeseries.png"
)

plt.savefig(
    timeseries_file,
    dpi=300,
    bbox_inches="tight"
)
```

```
plt.close()

#Save model inputes

model_data_file = os.path.join(
    output_folder,
    "XGB_swell_satellite_model_input_data.csv"
)

model_df.to_csv(
    model_data_file,
    index=False
)

#Outputs
print("\nSaved outputs:")
print(metrics_file)
print(predictions_file)
print(removed_rows_file)
print(importance_file)
print(importance_plot_file)
print(scatter_file)
print(timeseries_file)
print(model_data_file)

if os.path.exists(tree_plot_file):
    print(tree_plot_file)

print("\nDone.")

#[1] XGBoost Developers, “Scikit-Learn estimator interface examples,”
#XGBoost Documentation. [Online]. Available:
#https://xgboost.readthedocs.io/en/stable/python/examples/sklearn_parallel.html
#[Accessed: May 28, 2026].
```

B. FNN Code

```
# FNN Chlorophyll Model with Environmental + Swell + Satellite Input
import os
import pandas as pd
import numpy as np
```

```
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping


# File paths

base_path = r"C:\Users\jorda\Documents\Masters
program\Thesis\Raw_data_tables\Finalized_data\Master_files"

input_file = "MASTER_with_satellite_and_clean_swell_30min.csv"
file_path = os.path.join(base_path, input_file)

output_folder = os.path.join(base_path, "FNN_swell_satellite_outputs")
os.makedirs(output_folder, exist_ok=True)

print("Does input file exist?", os.path.exists(file_path))
print("Output folder:", output_folder)


# Load data

df = pd.read_csv(file_path)

print("\nOriginal rows:")
print(len(df))

print("\nColumns:")
print(df.columns.tolist())


# Features and targets

features = [
    # Environmental variables
    "WindSpeed",
    "WindGust",
```

```
    "WindDirection",
    "AirTemperature",
    "RelativeHumidity",
    "AtmosphericPressure",
    "SolarRadiation",
    "Rainfall",
    "DewPoint",
    "sea_water_practical_salinity",

    # Swell variables
    "sea_water_temperature_SWELL",
    "sea_surface_wave_mean_period_SWELL",
    "sea_surface_wave_period_at_variance_spectral_density_maximum_SWELL",
    "sea_surface_wave_significant_height_SWELL",
    "sea_surface_wave_from_direction_SWELL",

    # Satellite baseline chlorophyll variables
    "Sentinel3_OpenOcean_Chl_FilledForward",
    "Sentinel3_OpenOcean_Age_hr",
    "Sentinel3_ComplexWaters_Chl_FilledForward",
    "Sentinel3_ComplexWaters_Age_hr"
]

target = "mass_concentration_of_chlorophyll_in_sea_water"
time_col = "TimeUTC_30min"


# Check each column in CSV

required_cols = features + [target, time_col]

missing_cols = [col for col in required_cols if col not in df.columns]

if missing_cols:
    raise ValueError(f"Missing required columns: {missing_cols}")


# Clean data
#(Some of these varriables in the clean data section were made using Codex suggestion)
model_df = df[required_cols].copy()
model_df[time_col] = pd.to_datetime(model_df[time_col], errors="coerce")

#this cleaning works the same as in the XGBoost script
for col in features + [target]:
```

```
    model_df[col] = pd.to_numeric(model_df[col], errors="coerce")

rows_before = len(model_df)

removed_rows = model_df[model_df[features + [target]].isna().any(axis=1)].copy()
model_df = model_df.dropna(subset=features + [target])

rows_after = len(model_df)

print("\nRows before model cleaning:")
print(rows_before)

print("\nRows removed due to missing selected features or target:")
print(rows_before - rows_after)

print("\nRows used for model:")
print(rows_after)

removed_rows_file = os.path.join(output_folder, "FNN_removed_rows_missing_inputs.csv")
removed_rows.to_csv(removed_rows_file, index=False)


# This model usese a training/testing split of 80/20 which is the same for XGBoost
X = model_df[features].values
y = model_df[target].values

X_train, X_test, y_train, y_test, time_train, time_test = train_test_split(
    X,
    y,
    model_df[time_col],
    test_size=0.2,
    random_state=42
)


# SCALE INPUTS
#Built in scaling is used so that various magnitudes for the inputs are
# not given more weight in the model where xscaled=(x−μ)/σ
scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
# FNN model architecture
#This architecture was adapted for my project from keras.io developer guides[1]
model = Sequential([
    #layer 1
    Dense(128, activation="relu", input_shape=(X_train_scaled.shape[1],)),
    Dropout(0.20),

    #layer 2
    Dense(64, activation="relu"),
    Dropout(0.20),

    #layer 3
    Dense(32, activation="relu"),

    Dense(1)
])

model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
    loss="mse",
    metrics=["mae"]
)

print("\nModel summary:")
model.summary()


#Early end
early_stop = EarlyStopping(
    monitor="val_loss",
    patience=25,#number of epochs without improvement before stopping
    restore_best_weights=True #return to best epoch weight
)


#Train
history = model.fit(
    X_train_scaled,
    y_train,
    validation_split=0.2,#reserve 20% training data for validation
    epochs=300, #number of complete passes through training data allowed
    batch_size=64,
    callbacks=[early_stop],
    verbose=1
```

```
)

#training summary
#this gives info about the number of epochs that were actually gone though
#along with training and validation loss
actual_epochs = len(history.history["loss"])

print("\n==============================")
print("TRAINING DETAILS")
print("==============================")

print("Actual epochs completed:", actual_epochs)
print("Maximum epochs allowed:", 300)

if actual_epochs < 300:
   print("Training stopped early.")
else:
   print("Training reached max epochs.")

print("Best validation loss:")
print(min(history.history["val_loss"]))

print("Final training loss:")
print(history.history["loss"][-1])

print("Final validation loss:")
print(history.history["val_loss"][-1])

# Predict
y_pred = model.predict(X_test_scaled).flatten()
y_train_pred = model.predict(X_train_scaled).flatten()


# Analysis
mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)
bias = np.mean(y_pred - y_test)

train_r2 = r2_score(y_train, y_train_pred)
test_r2 = r2

print("FNN using swell, env., and satellite")
print(f"MAE:  {mae:.4f}")
```

```
print(f"RMSE: {rmse:.4f}")
print(f"R²:   {r2:.4f}")
print(f"Bias: {bias:.4f}")

print("Overfitting Check")
print(f"Train R²: {train_r2:.4f}")
print(f"Test  R²: {test_r2:.4f}")
print(f"R² Gap:   {train_r2 - test_r2:.4f}")


# Save metrics
metrics_df = pd.DataFrame({
    "Metric": ["MAE", "RMSE", "R2", "Bias", "Train_R2", "Test_R2", "R2_Gap"],
    "Value": [mae, rmse, r2, bias, train_r2, test_r2, train_r2 - test_r2]
})

metrics_file = os.path.join(output_folder, "FNN_swell_satellite_metrics.csv")
metrics_df.to_csv(metrics_file, index=False)


# Save prediction
results_df = pd.DataFrame({
    "TimeUTC_30min": time_test.values,
    "Actual_Chlorophyll": y_test,
    "Predicted_Chlorophyll": y_pred,
    "Residual": y_pred - y_test
})

results_df = results_df.sort_values("TimeUTC_30min")
#Save to seperate file(codex was used to write file saving code here)
predictions_file = os.path.join(output_folder, "FNN_swell_satellite_predictions_vs_actual.csv")
results_df.to_csv(predictions_file, index=False)


# Save model input data
model_data_file = os.path.join(output_folder, "FNN_swell_satellite_model_input_data.csv")
model_df.to_csv(model_data_file, index=False)


# Plot training loss (codex wat used for formatting of the plots here)
plt.figure(figsize=(8, 6))
plt.plot(history.history["loss"], label="Training Loss")
plt.plot(history.history["val_loss"], label="Validation Loss")
plt.xlabel("Epoch")
```

```
plt.ylabel("MSE Loss")
plt.title("FNN Training and Validation Loss")
plt.legend()
plt.grid(True)

loss_plot_file = os.path.join(output_folder, "FNN_training_validation_loss.png")
plt.savefig(loss_plot_file, dpi=300, bbox_inches="tight")
plt.close()


# Plot predicted vs. actual (codex was used for plot formating here)
plt.figure(figsize=(8, 8))
plt.scatter(y_test, y_pred, alpha=0.5)

min_val = min(y_test.min(), y_pred.min())
max_val = max(y_test.max(), y_pred.max())

plt.plot([min_val, max_val], [min_val, max_val], linestyle="--")

plt.xlabel("Actual Chlorophyll")
plt.ylabel("Predicted Chlorophyll")
plt.title("FNN with Satellite: Predicted vs Actual Chlorophyll")
plt.grid(True)

scatter_file = os.path.join(output_folder, "FNN_swell_satellite_predicted_vs_actual.png")
plt.savefig(scatter_file, dpi=300, bbox_inches="tight")
plt.close()


# Plots time series
#This is the same style as the XGBoost with predicted in orange and in-situ in blue
plt.figure(figsize=(14, 6))
plt.plot(results_df["TimeUTC_30min"], results_df["Actual_Chlorophyll"], label="Actual",
linewidth=1)
plt.plot(results_df["TimeUTC_30min"], results_df["Predicted_Chlorophyll"], label="Predicted",
linewidth=1)

plt.xlabel("Time")
plt.ylabel("Chlorophyll")
plt.title("FNN with Satellite: Actual vs Predicted Over Time")
plt.legend()
plt.grid(True)
plt.xticks(rotation=45)
```

```
timeseries_file = os.path.join(output_folder,
"FNN_swell_satellite_actual_vs_predicted_timeseries.png")
plt.savefig(timeseries_file, dpi=300, bbox_inches="tight")
plt.close()


# Plot zoomed in time series (Codex used for fomating)
# Use two distinct date windows

zoom_windows = [
    ("2023-06-01", "2023-07-31", "summer_2023_zoom"),
    ("2024-01-01", "2024-02-29", "winter_2024_zoom")
]

for start_date, end_date, label in zoom_windows:

    zoom_df = results_df[
        (results_df["TimeUTC_30min"] >= start_date) &
        (results_df["TimeUTC_30min"] <= end_date)
    ].copy()

    if zoom_df.empty:
        print(f"No data found for {label}: {start_date} to {end_date}")
        continue

    plt.figure(figsize=(15, 6))

    plt.plot(
        zoom_df["TimeUTC_30min"],
        zoom_df["Actual_Chlorophyll"],
        label="Actual",
        linewidth=1.5
    )

    plt.plot(
        zoom_df["TimeUTC_30min"],
        zoom_df["Predicted_Chlorophyll"],
        label="Predicted",
        linewidth=1.5
    )

    plt.xlabel("Time")
    plt.ylabel("Chlorophyll")
    plt.title(f"FNN Actual vs Predicted Chlorophyll: {start_date} to {end_date}")
```

```
    plt.legend()
    plt.grid(True)
    plt.xticks(rotation=45)

    zoom_file = os.path.join(
        output_folder,
        f"FNN_actual_vs_predicted_timeseries_{label}.png"
    )

    plt.savefig(zoom_file, dpi=300, bbox_inches="tight")
    plt.close()

    print("Saved zoomed time series:", zoom_file)

#harmful algal bloom (HAB) region for reference to motivation
#of understanding chlorophyl levels
hab_threshold = 10.0  # ug/L

##the HAB regions occur above 10.0 ug/l
results_df["HAB_Actual"] = results_df["Actual_Chlorophyll"] >= hab_threshold

plt.figure(figsize=(15, 6))

# Plot the actual and predicted chlorophyll (codex used for plot formatting)
plt.plot(
    results_df["TimeUTC_30min"],
    results_df["Actual_Chlorophyll"],
    label="Actual Chlorophyll",
    linewidth=1.5
)

plt.plot(
    results_df["TimeUTC_30min"],
    results_df["Predicted_Chlorophyll"],
    label="Predicted Chlorophyll",
    linewidth=1.5
)

# This adds the horizontal HAB threshold line
plt.axhline(
    y=hab_threshold,
    linestyle="--",
    linewidth=1.5,
    label=f"HAB Threshold = {hab_threshold} ug/L"
```

```
)

# Shade regions where actual chlorophyll exceeds threshold
plt.fill_between(
    results_df["TimeUTC_30min"],
    results_df["Actual_Chlorophyll"],
    hab_threshold,
    where=results_df["Actual_Chlorophyll"] >= hab_threshold,
    alpha=0.25,
    label="Measured HAB Region"
)

plt.xlabel("Time")
plt.ylabel("Chlorophyll (ug/L)")
plt.title("FNN Predicted vs Actual Chlorophyll with HAB Regions")
plt.legend()
plt.grid(True)
plt.xticks(rotation=45)

hab_plot_file = os.path.join(
    output_folder,
    "FNN_predicted_vs_actual_with_HAB_regions.png"
)

plt.savefig(
    hab_plot_file,
    dpi=300,
    bbox_inches="tight"
)

plt.close()

print(hab_plot_file)

model_file = os.path.join(output_folder, "FNN_swell_satellite_model.keras")
model.save(model_file)


# Outputs
print("\nSaved outputs:")
print(metrics_file)
print(predictions_file)
print(removed_rows_file)
print(loss_plot_file)
```

```
print(scatter_file)
print(timeseries_file)
print(model_data_file)
print(model_file)

print("\nDone.")

#[1] Keras Team, “The Sequential model,” Keras Documentation. [Online].
# Available: https://keras.io/guides/sequential_model/. [Accessed: Jun. 6, 2026].
```

C. Supporting figures Code

a. Figure 5.6: Data availability (MATLAB)

```
clear; clc; close all;
% Graph of data availability for input features and in-situ chlorophyll
% File: UNCLEANED_time_aligned_master_30min.csv
filePath = "C:\Users\jorda\Documents\Masters
program\Thesis\Raw_data_tables\Finalized_data\Uncleaned input
files\UNCLEANED_time_aligned_master_30min.csv";
[outFolder, fileBase, ~] = fileparts(filePath);
outFolder = fullfile(outFolder, "Selected_input_availability_plots");
if ~exist(outFolder, "dir")
  mkdir(outFolder);
end
fprintf("Reading:\n%s\n", filePath);
%readtable is a built in MATLAB function which imports csv data
T = readtable(filePath, "PreserveVariableNames", true);
%find columns from the data
varNames = string(T.Properties.VariableNames);
timeCandidates = varNames( ...
  contains(lower(varNames), "time") | ...
  contains(lower(varNames), "date"));
if isempty(timeCandidates)
  error("No time/date column found.");
end
timeCol = timeCandidates(1);
fprintf("Using time column: %s\n", timeCol);
Time = T.(timeCol);
%converts strings into matlab datetime objects
% that can be used for chronilogical sorting and filtering
if ~isdatetime(Time)
  Time = datetime(Time, "TimeZone", "UTC");
end
Time.TimeZone = "";
```

```
validTime = ~isnat(Time);
Time = Time(validTime);
T = T(validTime, :);
T.(timeCol) = [];
[Time, sortIdx] = sort(Time);
T = T(sortIdx, :);
[Time, uniqueIdx] = unique(Time);
T = T(uniqueIdx, :);
%define each varriabl
environmentalExpected = [
  "WindSpeed"
  "WindGust"
  "WindDirection"
  "AirTemperature"
  "RelativeHumidity"
  "AtmosphericPressure"
  "SolarRadiation"
  "Rainfall"
  "DewPoint"
  "sea_water_practical_salinity"
];
swellExpected = [
  "sea_water_temperature_SWELL"
  "sea_surface_wave_mean_period_SWELL"
  "sea_surface_wave_period_at_variance_spectral_density_maximum_SWELL"
  "sea_surface_wave_significant_height_SWELL"
  "sea_surface_wave_from_direction_SWELL"
];
satelliteExpected = [
  "Sentinel3_OpenOcean_Chl_FilledForward"
  "Sentinel3_OpenOcean_Age_hr"
  "Sentinel3_ComplexWaters_Chl_FilledForward"
  "Sentinel3_ComplexWaters_Age_hr"
];
%find matching columns for each input feature type and the in-situ values
tableVars = string(T.Properties.VariableNames);
tableVarsLower = lower(tableVars);
selectedVars = strings(0,1);
groupNames = strings(0,1);
% Environmental
for i = 1:length(environmentalExpected)
  target = environmentalExpected(i);
  targetLower = lower(target);
  exactIdx = find(tableVarsLower == targetLower, 1);
```

```
  if ~isempty(exactIdx)
    selectedVars(end+1,1) = tableVars(exactIdx);
    groupNames(end+1,1) = "Environmental";
  else
    partialIdx = find(contains(tableVarsLower, targetLower), 1);
    if ~isempty(partialIdx)
      selectedVars(end+1,1) = tableVars(partialIdx);
      groupNames(end+1,1) = "Environmental";
    else
      fprintf("Missing environmental variable: %s\n", target);
    end
  end
end
% Swell
for i = 1:length(swellExpected)
  target = swellExpected(i);
  targetLower = lower(target);
  exactIdx = find(tableVarsLower == targetLower, 1);
  if ~isempty(exactIdx)
    selectedVars(end+1,1) = tableVars(exactIdx);
    groupNames(end+1,1) = "Swell";
  else
    fprintf("Missing swell variable: %s\n", target);
  end
end
% Satellite
for i = 1:length(satelliteExpected)
  target = satelliteExpected(i);
  targetLower = lower(target);
  exactIdx = find(tableVarsLower == targetLower, 1);
  if ~isempty(exactIdx)
    selectedVars(end+1,1) = tableVars(exactIdx);
    groupNames(end+1,1) = "Satellite";
  else
    fprintf("Missing satellite variable: %s\n", target);
  end
end
fprintf("\nVariables included in plots:\n");
for i = 1:length(selectedVars)
  fprintf("%s - %s\n", groupNames(i), selectedVars(i));
end
if isempty(selectedVars)
  error("No selected variables were found.");
end
```

```
%create 30 minute grid (for each datapoint to fill as full or empty)
fullTime = (min(Time):minutes(30):max(Time))';
available = false(length(fullTime), length(selectedVars));
%Ismember is abulit in MATLAB function that matches observation timestamps
%to a 30 minute time grid
[isPresent, loc] = ismember(Time, fullTime);
dataMatrix = T{:, selectedVars};
available(loc(isPresent), :) = ~ismissing(dataMatrix(isPresent, :));
missing = ~available;
%clean the labels
plotLabels = strings(length(selectedVars), 1);
for i = 1:length(selectedVars)
  cleaname = selectedVars(i);
  cleanName = erase(cleanName, "_ENV");
  cleanName = erase(cleanName, "ENV");
  cleanName = strrep(cleanName, "_", " ");
  cleanName = strrep(cleanName, "Sentinel3", "Sentinel-3");
  cleanName = strrep(cleanName, "OpenOcean", "Open Ocean");
  cleanName = strrep(cleanName, "ComplexWaters", "Complex Waters");
  cleanName = strrep(cleanName, "SWELL", "");
  cleanName = regexprep(cleanName, '(?<=[a-z])(?=[A-Z])', ' ');
  cleanName = regexprep(cleanName, ' +', ' ');
  cleanName = strtrim(cleanName);
  plotLabels(i) = groupNames(i) + " - " + cleanName;
end
%PLot the availability of each input
figure("Position", [100 100 1800 950]);
%Bulitin MATLAB func that creates a heatmap like grid to show missing
%features
imagesc(fullTime, 1:length(selectedVars), available');
colormap(gca, [1 1 1; 0 0.4470 0.7410]);
colorbar("Ticks", [0 1], "TickLabels", ["Missing", "Available"]);
yticks(1:length(selectedVars));
yticklabels(plotLabels);
xlabel("Date");
ylabel("Selected Input Feature");
title("Selected Input Feature Availability Timeline", ...
  "Interpreter", "none");
set(gca, "YDir", "normal");
grid on;
hold on;
envEnd = sum(groupNames == "Environmental");
swellEnd = envEnd + sum(groupNames == "Swell");
if envEnd > 0 && envEnd < length(selectedVars)
```

```
  yline(envEnd + 0.5, "k-", "LineWidth", 1.5);
end
if swellEnd > 0 && swellEnd < length(selectedVars)
  yline(swellEnd + 0.5, "k-", "LineWidth", 1.5);
end
saveas(gcf, fullfile(outFolder, fileBase + "_selected_input_availability_timeline.png"));
% Plot of Percent available for each feature
percentAvailable = mean(available, 1) * 100;
figure("Position", [100 100 1200 900]);
%Built in matlab func which generates horizontal bar charts
barh(percentAvailable);
yticks(1:length(selectedVars));
yticklabels(plotLabels);
xlabel("Percent Available");
ylabel("Selected Input Feature");
title("Percent Available by Selected Input Feature", ...
  "Interpreter", "none");
xlim([0 100]);
grid on;
saveas(gcf, fullfile(outFolder, fileBase + "_selected_percent_available_by_input.png"));
fprintf("\nDone.\n");
fprintf("Plots saved to:\n%s\n", outFolder);
```

b. Figure 5.2-5.5: Sentinel-3 open Ocean and complex waters Chlorophyll estimate vs ground truth comparisons plot (MATLAB code)

```
%% Merge 30-min Ground Truth/Environmental Data with Satellite Chlorophyll
% Ground truth/env file:
%   Time = Column A (30-minute resolution)
%
% Satellite file:
%   Time = Column A
%   Chlorophyll = Column E
%   Sheet 1 = Open ocean
%   Sheet 2 = Complex waters
%
% Output:
%   Full 30-min dataset still saved
%   Comparison plots only use matched points within +/- 1 hour
clear; clc; close all;
%% File paths
basePath = "C:/Users/jorda/Documents/Masters
program/Thesis/Raw_data_tables/Finalized_data/";
```

```
groundFile = basePath + "Alligned_ground_truth_env_UTC.xlsx";
satFile    = basePath + "Chlorophyll_Sattelite_Data_UTC.xlsx";
outputFile = basePath +
"Alligned_ground_truth_env_with_satellite_chlorophyll_30min_FULL.csv";
%% Read files
GT = readtable(groundFile);
SAT_open    = readtable(satFile, "Sheet", "Open ocean");
SAT_complex = readtable(satFile, "Sheet", "Complex waters");
%% Extract and convert ground truth time
gtTimeRaw = GT{:,1};
gtTime = convertTimeColumn(gtTimeRaw);
gtTime.TimeZone = '';
%% Sort ground truth dataset
[gtTime, gtIdx] = sort(gtTime);
GT = GT(gtIdx,:);
%% Process both satellite sheets
[openFilled, openSourceTime, openAge_hr, openMatched] = ...
  processSatelliteDataset(GT, gtTime, SAT_open, "Sentinel-3 Open Ocean");
[complexFilled, complexSourceTime, complexAge_hr, complexMatched] = ...
  processSatelliteDataset(GT, gtTime, SAT_complex, "Sentinel-3 Complex Waters");
%% Add full filled-forward satellite columns to GT
GT.Sentinel3_OpenOcean_Chl_FilledForward = openFilled;
GT.Sentinel3_OpenOcean_Source_Time = openSourceTime;
GT.Sentinel3_OpenOcean_Age_hr = openAge_hr;
GT.Sentinel3_ComplexWaters_Chl_FilledForward = complexFilled;
GT.Sentinel3_ComplexWaters_Source_Time = complexSourceTime;
GT.Sentinel3_ComplexWaters_Age_hr = complexAge_hr;
%% Save full output
writetable(GT, outputFile);
fprintf("\nhere mrege the full dataset and its been saved to:\n%s\n", outputFile);
fprintf("the amount of rows in full dataset is: %d\n", height(GT));
%% Plot filled-forward satellite chlorophyll
figure;
plot(gtTime, openFilled, 'LineWidth', 1.5);
grid on;
xlabel('Time');
ylabel('Satellite Chlorophyll Retrieval Estimate (mg/m^3)');
title('Sentinel-3 Open Ocean Filled Forward Satellite Chlorophyll');
figure;
plot(gtTime, complexFilled, 'LineWidth', 1.5);
grid on;
xlabel('Time');
ylabel('Satellite Chlorophyll (mg/m^3)');
title('Sentinel-3 Complex Waters Filled Forward Satellite Retrieval Chlorophyll Estimate');
```

```
%% Plot satellite age
figure;
plot(gtTime, openAge_hr, 'LineWidth', 1.5);
grid on;
xlabel('Time');
ylabel('Satellite Data Age (hours)');
title('Sentinel-3 Open Ocean Satellite Data Age');
figure;
plot(gtTime, complexAge_hr, 'LineWidth', 1.5);
grid on;
xlabel('Time');
ylabel('Satellite Data Age (hours)');
title('Sentinel-3 Complex Waters Satellite Data Age');
%% Make matched-time comparison plots only
makeComparisonPlots(openMatched, "Sentinel-3 Open Ocean");
makeComparisonPlots(complexMatched, "Sentinel-3 Complex Waters");
%% Save matched datasets
open_matched_file = basePath + "Matched_OpenOcean_within_1hr.csv";
complex_matched_file = basePath + "Matched_ComplexWaters_within_1hr.csv";
writetable(openMatched, open_matched_file);
writetable(complexMatched, complex_matched_file);
fprintf("\nthe matched open-ocean points have been saved to:\n%s\n", open_matched_file);
fprintf("the matched complex-water points saved to:\n%s\n", complex_matched_file);
%% Local function to process one satellite dataset
function [filledSatChl, sat_source_time, sat_age_hr, matchedTable] = ...
  processSatelliteDataset(GT, gtTime, SAT, labelName)
  %% Extract satellite time and chlorophyll
  sat_time_raw = SAT{:,1};
  satChl = SAT{:,5};
  if ~isnumeric(satChl)
     satChl = str2double(string(satChl));
  end
  %% Convert satellite time
  satTime = convertTimeColumn(sat_time_raw);
  satTime.TimeZone = '';
  %% Clean satellite data
  validSAT = ~isnat(satTime) & ~isnan(satChl);
  satTime = satTime(validSAT);
  satChl  = satChl(validSAT);
  %% Sort satellite data
  [satTime, satIdxSort] = sort(satTime);
  satChl = satChl(satIdxSort);
  %% Filled-forward satellite values for full output file
  n = height(GT);
```

```
filledSatChl = nan(n,1);
sat_source_time = NaT(n,1);
sat_age_hr = nan(n,1);
satIdx = 1;
for i = 1 : n
   while satIdx < length(satTime) && satTime(satIdx+1) <= gtTime(i)
      satIdx = satIdx + 1;
   end
   if satTime(satIdx) <= gtTime(i)
      filledSatChl(i) = satChl(satIdx);
      sat_source_time(i) = satTime(satIdx);
      sat_age_hr(i) = hours(gtTime(i) - satTime(satIdx));
   end
end
%% Matched-pair dataset within +/- 1 hour only
groundChl = GT.mass_concentration_of_chlorophyll_in_sea_water;
matchedGT_time = NaT(length(satTime),1);
matched_Sat_Time = NaT(length(satTime),1);
matched_Ground_Chl = nan(length(satTime),1);
matched_Sat_Chl = nan(length(satTime),1);
timeDifference_hr = nan(length(satTime),1);
for j = 1:length(satTime)
   timeDiffs = abs(hours(gtTime - satTime(j)));
   [minDiff, gtMatchIdx] = min(timeDiffs);
   if minDiff <= 1
      gtVal = groundChl(gtMatchIdx);
      satVal = satChl(j);
      if ~isnan(gtVal) && ~isnan(satVal) && gtVal >= 0 && satVal >= 0
         matchCount = matchCount + 1;
         matchedGT_time(matchCount) = gtTime(gtMatchIdx);
         matched_Sat_Time(matchCount) = satTime(j);
         matched_Ground_Chl(matchCount) = gtVal;
         matched_Sat_Chl(matchCount) = satVal;
         timeDifference_hr(matchCount) = hours(gtTime(gtMatchIdx) - satTime(j));
      end
   end
end
matchedGT_time = matchedGT_time(1:matchCount);
matched_Sat_Time = matched_Sat_Time(1:matchCount);
matched_Ground_Chl = matched_Ground_Chl(1:matchCount);
matched_Sat_Chl = matched_Sat_Chl(1:matchCount);
timeDifference_hr = timeDifference_hr(1:matchCount);
matchedTable = table(matchedGT_time, matched_Sat_Time, timeDifference_hr, ...
   matched_Ground_Chl, matched_Sat_Chl);
```

```
  matchedTable.Properties.VariableNames = { ...
    'GroundTruth_Time', ...
    'Satellite_Time', ...
    'Time_Difference_hr', ...
    'GroundTruth_Chlorophyll', ...
    'Satellite_Chlorophyll'};
  fprintf("\n%s all of the matched points are within +/- 1 hour: %d\n", labelName, matchCount);
end
%% Local function for comparison and regression plots
function makeComparisonPlots(matchedTable, labelName)
  matched_time = matchedTable.GroundTruth_Time;
  matched_ground = matchedTable.GroundTruth_Chlorophyll;
  matched_sat = matchedTable.Satellite_Chlorophyll;
  %% Time comparison plot
  figure;
  plot(matched_time, matched_ground, '-o', 'LineWidth', 1.2);
  hold on;
  plot(matched_time, matched_sat, '-s', 'LineWidth', 1.2);
  grid on;
  xlabel('Time');
  ylabel('Chlorophyll Concentration (\mug/L or mg/m^3)');
  title("Matched-Time Satellite vs Ground Truth Chlorophyll - " + labelName);
  legend("Ground Truth Chlorophyll", labelName + " Satellite Chlorophyll", ...
    "Location", "best");
  %% Regression plot
  figure;
  scatter(matched_sat, matched_ground, 'x');
  hold on;
  grid on;
  p = polyfit(matched_sat, matched_ground, 1);
  xfit = linspace(min(matched_sat), max(matched_sat), 100);
  yfit = polyval(p, xfit);
  plot(xfit, yfit, 'LineWidth', 1.5);
  ypred = polyval(p, matched_sat);
  SS_res = sum((matched_ground - ypred).^2);
  SS_tot = sum((matched_ground - mean(matched_ground)).^2);
  R2 = 1 - SS_res/SS_tot;
  xlabel(labelName + " Satellite Chlorophyll");
  ylabel('Ground Truth Chlorophyll');
  title("Linear Regression: Ground Truth vs " + labelName + " Within 1 Hour");
  eqText = sprintf("y = %.3f x + %.3f\nR^2 = %.3f", p(1), p(2), R2);
  text(min(xfit), max(matched_ground)*0.9, eqText, 'FontSize', 12);
  legend('Data', 'Fit', 'Location', 'best');
  %% Print statistics
```

```
    RMSE = sqrt(mean((matched_sat - matched_ground).^2));
    MAE = mean(abs(matched_sat - matched_ground));
    bias = mean(matched_sat - matched_ground);
    R = corr(matched_sat, matched_ground, "Rows", "complete");
    fprintf("\n%s Results using matched points within +/- 1 hour:\n", labelName);
    fprintf("Matched points: %d\n", length(matched_ground));
    fprintf("RMSE: %.4f\n", RMSE);
    fprintf("MAE: %.4f\n", MAE);
    fprintf("Bias: %.4f\n", bias);
    fprintf("Pearson R: %.4f\n", R);
    fprintf("R^2: %.4f\n", R2);
end
%% Local function for time conversion
function timeOut = convertTimeColumn(timeRaw)
    if isdatetime(timeRaw)
        timeOut = timeRaw;
        timeOut.TimeZone = 'UTC';
        return;
    end
    if isnumeric(timeRaw)
        timeOut = datetime(timeRaw, 'ConvertFrom', 'excel', 'TimeZone', 'UTC');
        return;
    end
    timeStr = string(timeRaw);
    timeStr = strtrim(timeStr);
    timeStr = erase(timeStr, "Z");
    timeStr = replace(timeStr, "T", " ");
    timeStr = replace(timeStr, "/", "-");
    formats = [
        "yyyy-MM-dd HH:mm:ss"
        "yyyy-MM-dd HH:mm"
        "yyyy-MM-dd"
        "MM-dd-yyyy HH:mm:ss"
        "MM-dd-yyyy HH:mm"
        "MM-dd-yyyy"
        "dd-MM-yyyy HH:mm:ss"
        "dd-MM-yyyy HH:mm"
        "dd-MM-yyyy"
    ];
    for k = 1:length(formats)
        try
            timeOut = datetime(timeStr, ...
                'InputFormat', formats(k), ...
                'TimeZone', 'UTC');
```

```
            return;
        catch
        end
    end
    error("Could not convert time column.");
End
```