Open-Surfline: Leveraging Machine Learning and NOAA Buoy Data for Coastal Wave Forecasts

Larsen Weigle[1] Calvin Laughlin[1]

1. Introduction

Wave formation is a complex process influenced by factors such as winds, water temperature, tides, swells, and bathymetry. These elements work in tandem to create the undulating patterns on the ocean's surface known as waves. The ability to predict these waves carries significant implications for various industries. Accurate wave forecasts can optimize shipping routes, enhancing both safety and efficiency. Additionally, they can benefit the renewable energy sector by improving performance predictions for wave energy converters. While historical predictions relied on basic heuristics, recent advancements in technology have enabled more precise modeling.

Our study tackles the challenge of ocean wave prediction using machine learning techniques, advancing the successes of prior research in the field. The existing benchmark for wave prediction is the Simulating Waves Nearshore (SWAN) Model. Although this model is highly accurate, it requires substantial computational resources. Our methods aim to accurately forecast wave heights without incurring the significant computational costs associated with physics-based models.

To achieve this, we employ machine learning models, evaluating their ability to forecast wave heights with varying lead times using comprehensive data sets from NOAA buoys in Monterey Bay, California. By utilizing an array of features and model architectures, we aim to refine wave prediction methods, enhancing their accuracy and computational efficiency, and scrutinize the potential and limitations of these machine learning applications in wave forecasting.

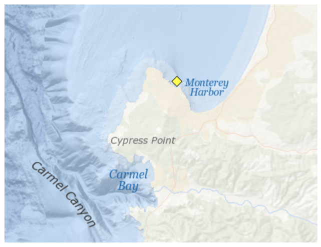

Figure 1. Location of buoy 46240 from NOAA.

2. Related Work

Wave height modeling has evolved significantly since its inception in the mid-20th century, with a notable shift occurring in the 1960s as researchers began to apply statistical methods to understand wave dynamics through the energy balance equation. This foundational work, which included contributions from Gelci et al. (5) and later refinements by Janssen (9) and Hasselmann (6), set the stage for subsequent advancements that would harness computational power to simulate complex wave interactions and the effects of wind and weather on wave patterns.

With the advent of more powerful computing in the 1980s, wave modeling entered a new phase characterized by detailed simulations of the sea surface, integrating comprehensive physical processes. These computational models, while precise, required significant resources, exemplified by the Simulating WAves Nearshore (SWAN) model developed by Booij et al. (2). SWAN's complex calculations, which encompassed a multitude of factors including wind effects and seabed interactions, represented a high watermark for accuracy but also highlighted the need for more efficient computational methods.

In response to the high computational cost of traditional models, machine learning approaches have recently been applied to wave height prediction, demonstrating potential for improved efficiency. James et al. (8) managed to reduce the computational load of SWAN using neural network techniques, and Miniuzzi et al. (12) achieved notable success with an LSTM model, attaining a 87% MAPE for a 6-hour forecast window. While their work indicated that additional features did not significantly enhance prediction accuracy, other studies, such as that by Berbić et al. (1), have found value in integrating wind data. These machine learning advancements present a compelling alternative to established physics-based models, offering similar accuracy with potentially reduced computational demands.

3. Dataset and Features

We utilized historical data from the National Data Buoy Center (NDBC), operated by the National Oceanic and Atmospheric Administration. The NDBC maintains 1322 stations across the Pacific and Atlantic oceans. Each buoy records a location-specific subset of standard meteorological data. Our focus was on Monterey Bay, CA, specifically Station 46240 near Cabrillo Point (see **figure 1**), selected for its comprehensive historical meteorological data.

Station 46240, a near-shore buoy, records various measurements such as wave height (WVHT), dominant wave period (DPD), average wave period (APD), and wave direction (MWD). Our dataset comprises hourly data from 2009 to 2022. To address sensor malfunctions, we identified null values indicated by numerical placeholders (99.99, 9999, 999, etc.), resulting in 319 rows with null values. These rows were excluded from the dataset. Additionally, we considered complete missing rows. During certain periods, no measurements were recorded, leading to non-sequential hourly data. We recorded the indices of these missing rows and applied a sliding window approach to construct our supervised learning dataset for training. We prepared three distinct groups datasets corresponding to forecasting lead times of 12, 18, and 24 hours. Despite experimenting with various window sizes and time discretizations, a window size of 48 with one-hour discretization proved most effective.

For dataset division, we strategically separated the data by year to prevent information leakage between the training, validation, and testing sets. The training set encompasses data from 2009 to 2018. The validation set includes the years 2019 and 2020, while the test set comprises data from 2021 and 2022. We utilized all four wave-related measurements — WVHT (Wave Height), DPD (Dominant Period), APD (Average Period), and MWD (Mean Wave Direction) — as our feature set. **Table 1** provides detailed information about these datasets.

Table 1. Resulting training datasets after processing

LEAD	FEATURES	PREDICTION	DIMENSIONS
12H	WVHT, DPD, ADP, MWD	WVHT	(96087 x 192)
18H	WVHT, DPD, ADP, MWD	WVHT	(96075 x 192)
24H	WVHT, DPD, ADP, MWD	WVHT	(96063 x 192)

We hypothesize that incorporating dominant wave period (DPD), average wave period (APD), and wave direction into our advanced models will enable them to discern more complex relationships between the 48-hour input measurements and the predicted wave height. These features are key components in established physics-based models where they are explicitly modeled alongside wave heights. Consequently, we anticipate that neural network architectures could benefit from data encapsulating these intricate relationships. However, it's noteworthy that Miniuzzi et al. (12) observed limited predictive power in features beyond wave height. Echoing this, Makarynskyy et al. (11) adopted a widely recognized approach in the field, relying solely on previous wave height measurements as a feature.

Finally, depending on the model, we normalized each feature to reconcile differences in their scales and units. This step proved crucial for only the multi-layer perception; omitting it led to a significant decline in its forecasting performance.

4. Methods

We apply a sliding window approach to transform time series data into a format suitable for supervised learning. We utilize a window size of 48 hours with hourly time discretization. For each instance x_i, we compile a sequence of 48 measurements spaced one hour apart. We take careful steps to confirm that these measurements are sequential and consistently spaced. The composition of each measurement contains the four wave related features (WVHT, DPD, APD, MWD) from the buoy.

To determine the corresponding y_i for each x_i, we identify the WVHT value that occurs at a specified lead time after the final measurement in x_i. Subsequently, we flatten each x_i into a one-dimensional vector in $\mathbb{R}^{1\times d}$, where d represents the product of the number of measurements and the number of features per measurement. Across all experiments, no data is shuffled to ensure temporal dependencies are preserved.

All models in our study utilize mean squared error (MSE) as the loss function, prioritizing the accurate prediction of crucial wave height spikes by heavily penalizing larger errors, which is essential for applications like navigation and coastal management.

4.1. Linear Regression

In adapting linear regression to our wave height prediction, we employed the standard formula:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n + \epsilon$$

where Y is the wave height, β_i are the model coefficients, and ϵ is the error term. To accommodate our dataset with 4 features across a 48-hour window, the model was extended to include 192 coefficients, theoretically allowing it to capture the temporal dynamics within each feature set. We trained and validated the model on our datasets, assessing its performance across various lead times to determine its efficacy in wave height forecasting.

4.2. XGBoost

For XGBoost, we leveraged its sequential tree-building approach, where each tree is designed to correct the residuals of its predecessors. The objective function optimized during training is given by:

$$Obj(\theta) = \ell(\theta) + \Omega(\theta)$$

where θ are the model parameters, $\ell(\theta)$ is the loss function reflecting the discrepancy between predictions and actual observations, and $\Omega(\theta)$ is the regularization term which mitigates overfitting. Our modification involved tuning XGBoost to handle the spatiotemporal aspects of our data, focusing on wave height prediction accuracy while preventing over-complexity in the model structure.

4.3. Multi-Layer Perceptron

The Multi-Layer Perceptron (MLP), a type of artificial neural network, has been customized for our task of forecasting wave heights. Our MLP architecture includes an input layer, two hidden layers — with the first containing 64 neurons and the second 32 neurons — employing ReLU (Rectified Linear Unit) activation functions, and a single-neuron output layer designed for regression. The input layer takes in a feature vector from buoy data that includes variables such as wave direction and wave periods. Through the hidden layers, the MLP learns to identify patterns within the data, while the output layer is responsible for generating continuous wave height predictions for the next 48 hours.

4.4. Long Short-Term Memory

LSTMs are a variation of Recurrent Neural Networks (RNNs) specifically designed for learning and remembering long-term dependencies in sequential data (7). LSTMs are relevant when it comes to wave prediction because they take into account both long-term and short-term dependencies, making them useful for predicting wave height based on hourly observations from buoys.

LSTMs use a cell state and gates to create a memory over sequences. The gates include a forget gate, an input gate, and an output gate. These gates coordinate information flow and enable the network to store or discard data over extended periods of time. The cell state is the memory layer that contains the context across the inputs. The forget gate decides whether we should remove previous memories based on new input from the network. The input gate decides whether we should update the previous memory of the network due to our new input. And the output gate is composed of the current input, the previous layer's output, and the persisted memory cell.

In time-series forecasting applications, LSTMs are particularly useful because of their ability to remember past information (in our case, previous wave heights and periods) over long sequences. Other models, such as linear regression, end up predicting based on just the last few examples, whereas LSTMs are adept at understanding patterns and dynamics that make future predictions more accurate.

To summarize, we illustrate the mathematical representations of each gate in the LSTM for forward propagation. X will denote our input, H' our previous layer's output, and H the output to be fed to the next layer

$$\begin{aligned}
\text{Forget Gate: } & F = \sigma(u_1 X + w_1 H') \\
\text{Input Gate: } & I = \sigma(u_2 X + w_2 H') \\
\text{Candidate State: } & G = \tanh(u_3 X + w_3 H') \\
\text{Memory State: } & C = C'F + IG \\
\text{Output State: } & O = \sigma(u_4 X + w_4 h') \\
\text{Output Hidden State: } & H' = O \tanh(C) \\
\text{Output: } & vH'
\end{aligned}$$

5. Experiments

5.1. Linear Regression Experiments

We established a baseline using linear regression, against which we could gauge the performance of more complex models. Utilizing wave height (WVHT), dominant wave period (DPD), average wave period (APD), and wave direction (MWD) as features, we trained our model on data from 2009 to 2018 and assessed it on test data from 2019 to 2020. We experimented with normalized and non-normalized data, and for this particular model architecture it seemed that non-normalized performed better. The MSEs for each lead time (non-normalized) are detailed in **Table 2**.

Table 2. Mean Squared Errors for Different Lead Times

Lead Time (Hours)	Mean Squared Error (MSE)
12	0.0633
18	0.0881
24	0.1068

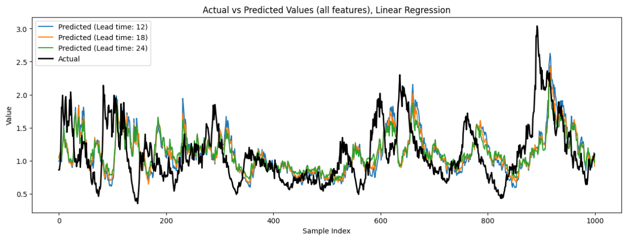

Figure 2. Actual vs predicted validation set values for linear regression.

The MSE values initially appeared promising. However, upon examining the graph (refer to **Figure 2**), it became evident that our model was experiencing ”lagging,” a common challenge in machine learning with time series data. This issue was identifiable by the model's predicted shape closely mirroring the actual shape, but shifted forward by several time steps (as shown in the graph). This typically occurs when the model excessively relies on the most recent values in the time series data for making predictions, essentially replicating the preceding trend rather than accurately forecasting future events.

In an attempt to address this, we experimented with various modifications, including altering the window, step, and lead times. Through this process, we identified that a window size of 48 and a step size of 1 yielded the most reliable outcomes. Ultimately, we concluded that linear regression was overly simplistic for time forecasting with our dataset. Its inability to recognize long-term dependencies or patterns was a significant limitation. Consequently, we shifted our focus to exploring other models that might more effectively capture the underlying relationships in the data.

5.2. XGBoost Experiments

Transitioning from linear regression, we turned to the XGBoost regressor, anticipating that its complex, non-linear approach would more effectively discern cycles, patterns, and long-term dependencies in our data. XGBoost was selected over other non-linear models for its robustness against overfitting and the flexibility of fine-tuning its hyperparameters to optimize model performance.

The XGBoost model was trained for different lead times of 12, 18, 24 hours. Model accuracy was evaluated using the mean squared error. We experimented with normalized and non-normalized data, and for this particular model architecture it also seemed that non-normalized performed better. After experimentation and analysis of validation MSEs, we determined the optimal hyperparameters for our model: n_estimators at approximately 1000, max_depth at 6, eta (learning rate) at 0.1, and both subsample and colsample_bytree set to 0.8. As indicated in **Table 3**, there was a slight increase in MSE.

Table 3. Mean Squared Errors for Different Lead Times

Lead Time (Hours)	Mean Squared Error (MSE)
12	0.0679
18	0.0939
24	0.1162

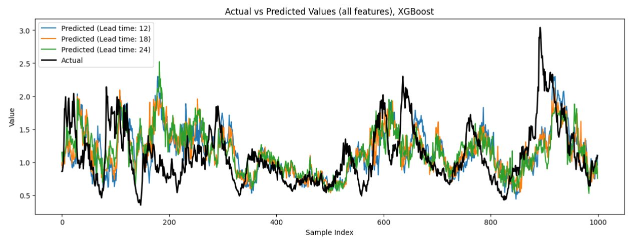

Figure 3. Actual vs predicted validation set values for XGBoost.

Despite the increase in MSE, **Figure 3** reveals that the model is actively attempting to forecast rather than merely predicting on the last value in each window, as was the case with linear regression. This approach provides more insightful predictions, as the model now evaluates the entire temporal dataset. However, it occasionally overextends in its predictions, as exemplified at index 600, where it predicts an increasing wave size contrary to the shrinking trend observed in the validation data. This behavior underscores the need for a model specifically tailored to handle temporal dependencies and time sequence data for more accurate forecasting.

5.3. Multi-Layer Perceptron Experiments

We implemented our Multi-layer Perceptron (MLP) using PyTorch and conducted experiments with architectures comprising 2 and 3 hidden layers. For each layer, we tested configurations of 128, 64, or 32 neurons. Ultimately, we opted for a 2-hidden-layer MLP. This decision was based on our observation that increasing the network size led to overfitting on the training set, evidenced by a spike in validation loss after approximately 20 epochs. We use an Adam Optimizer with a weight decay of 1e-5, a learning rate of 1e-4, and a drop out rate of 0.1. Furthermore, we evaluated the MLP's performance with both normalized and unnormalized inputs, finding that the model showed significantly improved performance with normalized data.

Table 4. Mean Squared Errors for Different Lead Times

Lead Time (Hours)	Mean Squared Error (MSE)
12	0.0655
18	0.0901
24	0.1124

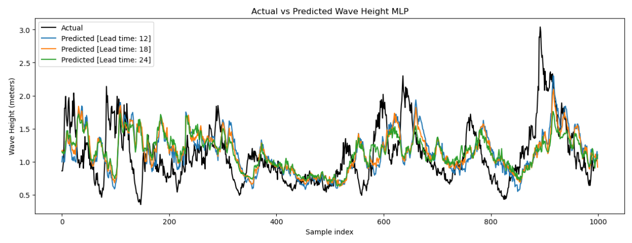

Figure 4. Actual vs predicted validation set values for the MLP.

The MLP displayed a "lagging" behavior akin to that of linear regression, as shown in the plot comparing predicted and actual wave heights in the validation dataset. This pattern suggests that the MLP struggled to capture the true relationship between the input features and future wave heights, instead reverting to replicating recent trends from the training examples. It's important to note that since we normalized the input features, including wave height, we inverted the predictions to their non-normalized counterparts in order to calculate the MSE.

5.4. Long Short-Term Memory Experiments

Our implementation of Long Short-Term Memory (LSTM) utilizes the PyTorch deep learning framework, which is well-suited for handling large datasets. The architecture of our LSTM model is as follows:

1. An input LSTM layer with 50 units. This balance between complexity and overfitting allows the layer to return sequences for the next layer.
2. A dropout layer to reduce overfitting, randomly deselecting a fraction of the input units.
3. A second LSTM layer, also with 50 units, which does not return sequences but instead the final state of the LSTM cells.
4. Another dropout layer, further mitigating overfitting.
5. A dense layer designed for multi-step forecasting. The number of neurons is determined by the required lead time. This setup enables the model to predict multiple future steps in a single forward pass.

For optimization, we use the Adam optimizer and employ the mean squared error as our loss function. The model is trained over 20 epochs, which we found to be optimal for reducing overfitting, with a batch size of 32. We experimented with normalized and non-normalized data, and for this particular model architecture it also seemed that non-normalized performed better.

This approach to LSTM modeling allows for multi-step predictions in a single forward pass. An alternative strategy, predicting each future step individually and compiling the predictions, tends to accumulate errors, especially in longer sequences. This error accumulation can lead to less accurate forecasts for longer lead times and fails to leverage the LSTM's ability to capture dependencies across multiple time steps.

Our results from training the LSTM on different lead times are seen in **Table 5** and **Figure 5**, indicating the model's capacity to learn from sequential data and make predictions based on patterns, cycles, and long-term dependencies:

Table 5. Mean Squared Errors for Different Lead Times

Lead Time (Hours)	Mean Squared Error (MSE)
12	0.0753
18	0.1066
24	0.1426

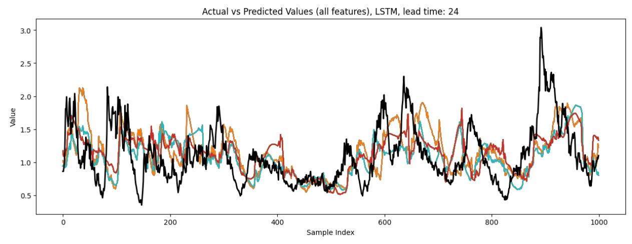

Figure 5. Results of LSTM on validation data
(black: validation, 12 hours: orange, 18 hours: blue, 24 hours: red)

Interestingly, around sample index 400, the model anticipates a significant spike in wave height, with predictions and actual values increasing simultaneously. This contrasts with basing predictions on momentum or the most recent data points, as seen in other models.

The LSTM, our best-performing model without lag issues, was applied to the test sets across various lead times. The final results showcased a Mean Squared Error (MSE) of 0.0782 for a 12-hour lead time, 0.1107 for an 18-hour lead time, and 0.1649 for a 24-hour lead time.

6. Conclusion

In conclusion, the task of predicting wave height using time series data has been challenging and presented a number of unforeseen difficulties. The most effective algorithm was the LSTM. Despite not achieving the lowest mean squared errors (MSEs), the LSTM proved to be valuable in forecasting. Unlike simpler models such as Linear Regression and XGBoost, it did not exhibit the lagging issue. The LSTM's ability to recognize and incorporate long-term patterns from time series data was key to its success.

For future work, we aim to implement additional measures to mitigate prediction lag, which is a prevalent issue in time series analysis. While we have explored various advanced models to reduce lag, other methods warrant investigation. For instance, the Neuro-Wavelet technique, as discussed in Dixit et al. (2015), offers a promising approach by decomposing data into approximate and detailed frequency components, thus reducing the correlation between sequential wave heights. Moreover, incorporating statistical features such as the rate of change of wave height could further diminish lag effects. Enhancing our model to adeptly handle such complexities will be a pivotal step toward more accurate and reliable wave height prediction systems.

7. Contributions

Larsen was responsible for the data processing pipeline and the MLP code. Calvin focused on the LR, XGBoost, and LSTM efforts. Both contributed equally to the creation of the paper.

References

[1] Jadran Berbić et al. Application of neural networks and support vector machine for significant wave height prediction. *Oceanologia*, 59(3):331–349, 2017.

[2] N. Booij, Roeland C. Ris, and Leo H. Holthuijsen. A third-generation wave model for coastal regions: 1. model description and validation. *Journal of Geophysical Research: Oceans*, 104(C4):7649–7666, 1999.

[3] Mercè Casas-Prat, Xiaolan L. Wang, and Joan P. Sierra. A physical-based statistical method for modeling ocean wave heights. *Ocean Modeling*, 73:59–75, 2014.

[4] Pradnya Dixit, Shreenivas Londhe, and Yogesh Dandawate. Removing prediction lag in wave height forecasting using neuro - wavelet modeling technique. *Ocean Engineering*, 93:74–83, 2015.

[5] R. Gelci. Prévision de la houle. la méthode des densités spectroangulaires. *Bull. Inform. Comité Central Oceanogr. d'Etude Côtes*, 9:416–435, 1957.

[6] Klaus Hasselmann. On the spectral dissipation of ocean waves due to white capping. *Boundary-Layer Meteorology*, 6:107–127, 1974.

[7] Sepp Hochreiter and J"urgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[8] Scott C. James, Yushan Zhang, and Fearghal O'Donncha. A machine learning framework to forecast wave conditions. *Coastal Engineering*, 137:1–10, 2018.

[9] Peter AEM Janssen. Wave-induced stress and the drag of air flow over sea waves. *Journal of Physical Oceanography*, 19(6):745–754, 1989.

[10] Peter AEM Janssen. Progress in ocean wave forecasting. *Journal of Computational Physics*, 227(7):3572–3594, 2008.

[11] O. Makarynskyy. Improving wave predictions with artificial neural networks. *Ocean Engineering*, 31(5-6):709–724, 2004.

[12] Felipe C. Minuzzi and Leandro Farina. A deep learning approach to predict significant wave height using long short-term memory. *Ocean Modelling*, 181:102151, 2023.

(1) (2) (3) (4) (6) (7) (9) (10) (8) (12) (11)