SEA LEVEL RISE ANALYSIS

INDEX

Team Members:
Varun Reddy
Dnyaneshwari Rakshe
Tanvi Nimbalkar

Link to GitHub Folder - https://github.com/VarunnReddyy/dmproject
Link to Website - https://dmproject1.streamlit.app/

Abstract:

Our major goal behind working on this project is to predict tidal levels and understand the sea-level variations using a comprehensive dataset of around 19 years from 1980 to 2025. We focused on deriving insights from various features with tidal metrics such as MHHW, MHW, and MSL. We started with data preprocessing by imputing missing values using mean and mode strategies, feature engineering to extract temporal patterns, and cyclical encoding for time-based data. Additionally, we processed numerical features using z-score normalization in order to ensure scale uniformity, and then we split the dataset into training, validation, and testing sets so that we can optimize model robustness. We plot the data visualizations using a variety of charts like time series plots, scatterplots, and correlation heatmaps. This helped us in highlighting critical patterns like rising sea levels and interdependencies among different tidal metrics. Moreover, we implemented different machine learning and deep learning models like Decision Trees, Random Forests, XGBoost combined with Prophet, LSTM with Dense layers, Temporal Convolutional Networks (TCN), and TCN+LSTM, to predict tidal levels and identify seasonal and temporal variations. With these analysis and insights we focused on providing a foundation for developing predictive models in order to enhance coastal management. We believe that this project will help coastal and environmental agencies in disaster preparedness, and navigation safety. With the help of historical tidal data, we ensured that the project delivers actionable insights for policymakers and urban planners in order to mitigate risks from environmental changes, ensuring a sustainable future for vulnerable coastal areas.

Dataset Description:

This dataset used in the project represents variations in tides and sea levels through time, with recordings at specific time intervals in GMT. The key features include the Highest and Lowest water levels in a given time, and tidal datums like Mean Higher High Water (MHHW), Mean High Water (MHW), Mean Sea Level (MSL), Mean Tide Level (MTL), Mean Low Water (MLW), and Mean Lower Low Water (MLLW) in feet. These parameters describe the range of tides and changes in sea level over a 19-year period. While the temporal data was preprocessed by the extraction of components such as month and hour, encoding cyclical patterns with sine and cosine transformations. Numerical missing values were imputed with column means, while categorical values were imputed as the mode. Besides that, numerical features were standardized with z-score normalization for having similar scales. The data is split into training, validation, and testing sets to facilitate model training and testing in maintaining robustness without overfitting.

Objective:

The objective of this project, in real-time, is to develop a predictive model for the accurate forecasting of tidal levels and variations in sea levels that can be used as a vital tool for managing coasts, preparing for disasters, and ensuring navigation safety. Using the history of tidal data, the model will look for patterns and anomalies in water levels that offer important insights for mitigating risks from flooding, storm surges, and sea-level rise. This will be very helpful for the policymakers, urban planners, and environmental agencies to make better decisions that are evidence-based in protecting the most vulnerable coastal areas, optimizing infrastructure planning, and developing more effective early warning systems for extreme weather events.

Data Collection and Cleaning:

We started with scraping data from a relevant datasource in order to generate a comprehensive list of stations for data collection. We believed that generating a comprehensive list was a crucial task to get started with our project because it would help us to target data analysis with respect to different stations located in different areas. After we prepared the station list, we used an API reference to retrieve the required information from each identified station.

Stations:

sea_level_station_products

station_id	0	1	2	3	4	5	6	7	8	9	10	11	12
1611400													
1612340													
1612401													
1612480													
1615680													
1617433													
1617760													
1619910													
1630000													
1631428													

Dataset:

	Date	Time (GMT)	Highest	MHHW (ft)	MHW (ft)	MSL (ft)	MTL (ft)	MLW (ft)	MLLW (ft)	Lowest (ft)	Inf	station_id
2	1/1/1980	0:00	2.484	1.857	1.394	0.827	0.799	0.203	-0.052	-0.597	0	1611400
3	2/1/1980	0:00	2.782	1.926	1.539	0.942	0.925	0.312	0.108	-0.118	0	1611400
4	3/1/1980	0:00	2.103	1.686	1.434	0.889	0.879	0.325	0.213	-0.075	0	1611400
5	4/1/1980	0:00	2.142	1.811	1.486	0.928	0.907	0.328	0.184	-0.187	0	1611400
6	5/1/1980	0:00	2.172	1.673	1.214	0.682	0.653	0.082	-0.138	-0.518	0	1611400
7	6/1/1980	0:00	2.263	1.783	1.273	0.683	0.663	0.053	-0.217	-0.517	0	1611400
8	7/1/1980	0:00	2.543	2.064	1.614	0.994	0.974	0.344	0.102	-0.148	0	1611400
9	8/1/1980	0:00	2.493	2.064	1.713	1.093	1.083	0.453	0.223	0.033	0	1611400
10	9/1/1980	0:00	2.333	1.969	1.644	1.027	1.006	0.367	0.217	0.013	0	1611400
11	10/1/1980	0:00	2.283	1.844	1.473	0.873	0.863	0.253	0.144	-0.295	0	1611400
12	11/1/1980	0:00	2.474	1.854	1.43	0.873	0.848	0.266	0.056	-0.276	0	1611400
13	12/1/1980	0:00	2.802	2.103	1.654	1.102	1.073	0.492	0.243	-0.075	0	1611400
14	1/1/1981	0:00	2.792	2.142	1.654	1.063	1.033	0.423	0.135	-0.197	0	1611400
15	2/1/1981	0:00	2.602	2.149	1.742	1.129	1.111	0.479	0.285	-0.016	0	1611400
16	3/1/1981	0:00	2.474	1.985	1.673	1.07	1.058	0.443	0.305	0.062	0	1611400
17	4/1/1981	0:00	2.133	1.793	1.443	0.873	0.853	0.263	0.113	-0.197	0	1611400
18	5/1/1981	0:00	2.523	2.024	1.585	0.922	0.912	0.243	0.033	-0.348	0	1611400
19	6/1/1981	0:00	2.454	1.893	1.365	0.784	0.755	0.144	-0.138	-0.466	0	1611400
20	7/1/1981	0:00	2.523	1.913	1.385	0.784	0.755	0.125	-0.138	-0.518	0	1611400
21	8/1/1981	0:00	2.503	2.034	1.604	1.014	1.004	0.394	0.194	-0.075	0	1611400

Steps implemented during Data Cleaning and Data Processing:

Handling Missing Values:

Handling missing values was a critical preprocessing step that we considered so that we could ensure that the dataset is both complete and consistent for model implementation and training it for further predictions. As we know that missing values often arise due to data collection from multiple sources, incomplete records, or external factors. Presence of missing values in a dataset can lead to errors or biases that can negatively impact model performance.

For numerical features like tide metrics - MHHW (ft) or MLLW (ft)), we imputed missing values using the mean of the respective attributes. We decided to proceed with this approach because it proves to be effective when the data distribution is relatively symmetrical and free from extreme outliers. Using this approach of imputing mean values it helped us ensure that the imputed values represented the central tendency of the data, and preserved feature scaling and overall dataset integrity.

Then for categorical or non-numeric columns like Date, Time (GMT)), we filled up the missing values with the most frequent value i.e. the mode of the corresponding attribute. The mode helped us identify the most common category or value in order to ensure that the imputation process that we implemented aligns with the majority of the data. For example, if most records for Time (GMT) are at midnight (00:00), it is reasonable to replace missing time values with this common occurrence.

Therefore, in this way this dual-strategy approach for handling missing values for both numerical and non-numerical features helped us ensure that statistical integrity is maintained for numerical data and categorical consistency is preserved for categorical or non-numerical data.

Feature Engineering:

We considered feature engineering so that we can transform raw data into meaningful features and can improve model performance and its predictions. During this phase, we created new features, extracted useful information, and restructured the dataset in order to make it more suitable for working with machine learning models. We applied feature engineering as follows:

1. **Extracting Components:**
    - From Date: We extracted components like Month and Day in order to account for seasonal trends and monthly variations in tidal levels.
    - From Time (GMT): We extracted the hour of the day in order to capture daily periodicity, because we know that tides often follow predictable diurnal cycles.
2. **Cyclical Encoding:**
    - In the cyclical encoding phase, we encoded the features like Time (GMT) into cyclical features using sine and cosine transformations as shown below:
        - Sin_Hour = sin(2π * Time / 24)

        - Cos_Hour = cos(2π * Time / 24)
    - This transformation helped us ensure that the time representation is cyclic (e.g., 23:00 is closer to 00:00 than to 12:00). This becomes a critical aspect for distance-based models like KNN.

Feature Scaling:

Using feature scaling helped us ensure that the numerical features are on the same scale. This especially becomes important for implementing machine learning models that are sensitive to feature magnitudes.

1. We standardized numerical features like tide metrics - MHHW (ft) and MLLW (ft)) using z-score normalization. This technique helped us adjust the features to have a mean of 0 and a standard deviation of 1 in order to ensure uniform scaling across all the numerical attributes.
2. We considered implementing standardization particularly for models such as:
    - Distance-based models (KNN): This ensures that features contribute equally to distance calculations.
    - Neural networks (LSTM): This helps in preventing gradients from being dominated by features with larger magnitudes.

After applying feature scaling, we observed that the dataset became well-prepared for model training, and helped us in avoiding potential biases and ensuring effective feature representation for models that are sensitive to scale.

Splitting the Data:

Splitting the data is a critical step in the machine learning workflow so that we can evaluate model performance and ensure generalizability to new or unseen data.

Train-Test Split:

1. We divided the dataset into training and testing sets using an 80-20 split - Test: 20% and Training: 80%
2. We also ensured that no data leakage occurred between the training and testing phases in order to further maintain the integrity of the evaluation process.

Validation Data: For models like LSTM, we set a portion of the training data as a validation set. And we used this data for:

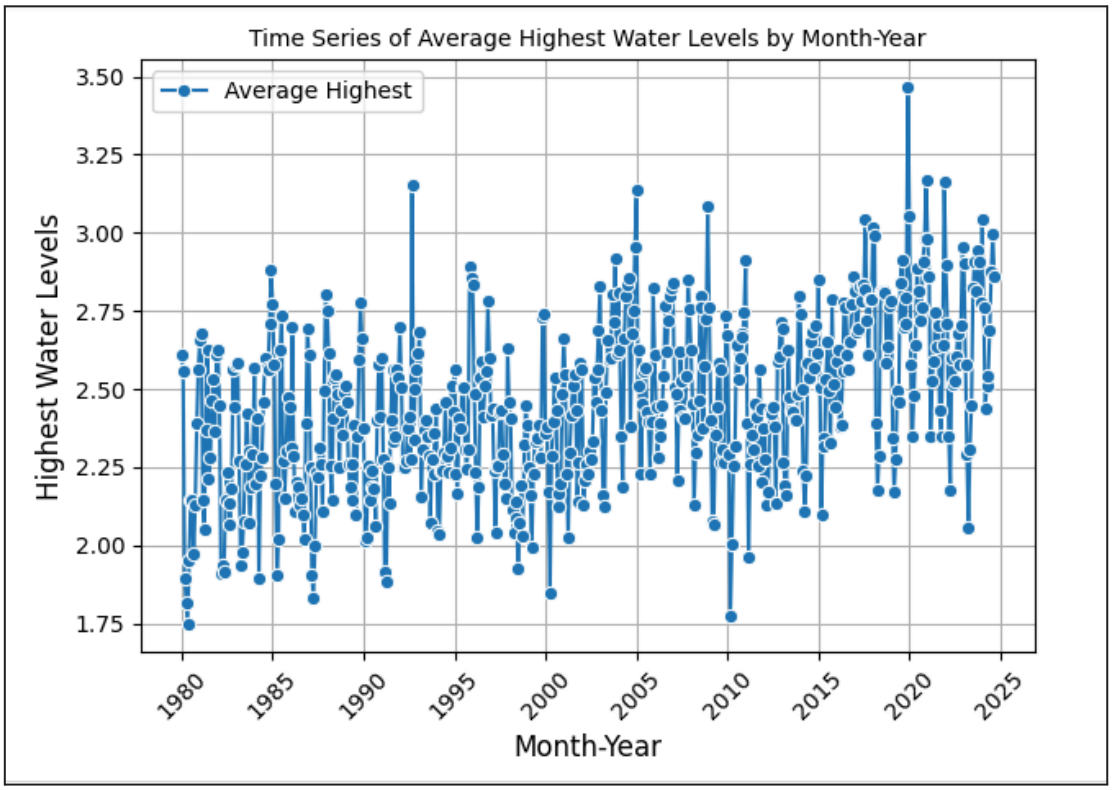
- Early stopping: In order to prevent overfitting by monitoring model performance during the training phase.
- Hyperparameter tuning: We used hyperparameter tuning to optimize the model parameters effectively.

In this way, by carefully splitting the data, we implemented robust model evaluation, and this allowed us to obtain accurate performance assessment and minimized the risk of overfitting or data contamination.

Data Visualisations:

We implemented a series of well-designed visualizations in order to effectively communicate the data patterns and trends that were observed during this project. We selected these visualizations so that we can effectively represent the key aspects of the dataset and provide data insights and understand the underlying relationships and patterns within the data. Each visualization highlights the most significant findings and offers us a detailed view of the attributes and their emerging trends.

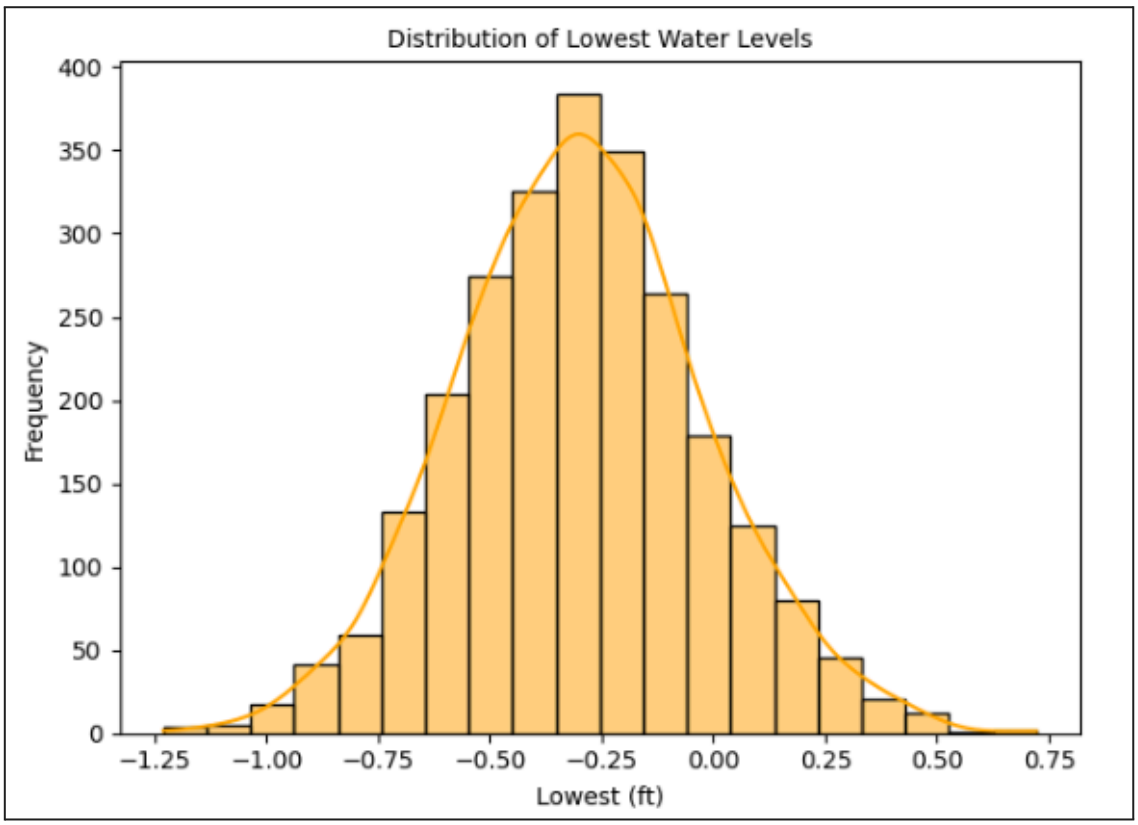
Time Series of the 'Highest' water levels:



Insights:

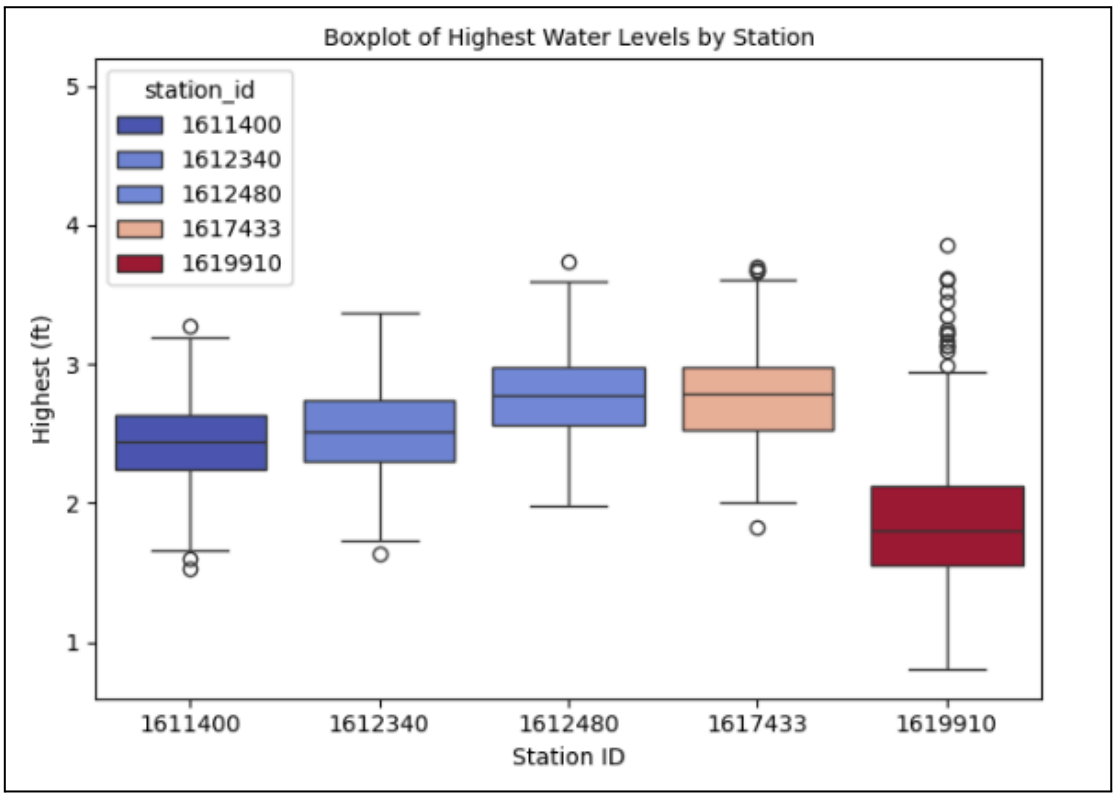
The time series visualization displayed above represents the variation in average highest water levels over time, from 1980 to 2025. This plot helps us understand a general upward trend which indicates that the highest tide levels have progressively increased over the years since 1980. Individual data points in the above plot represent both seasonal and irregular variations, and reflect periodic spikes and dips. In this way, this visualization helped us identify the trend that helps us in understanding potential changes in water level patterns observed over the years.

Histogram Distribution of 'Lowest' water levels:



Insights:
The histogram above displays the frequency distribution of 'Lowest' water levels, and shows us how often the specific ranges of water levels occur. We can observe that the majority of the data is concentrated around the mean value which is approximately around **-0.25 feet**, and the frequency values are tapering off symmetrically on either side. This helps us in identifying that most low water levels fall within a narrow range, while extreme values could be rarely observed. This reflects a balanced and predictable pattern in the dataset's lower bounds.

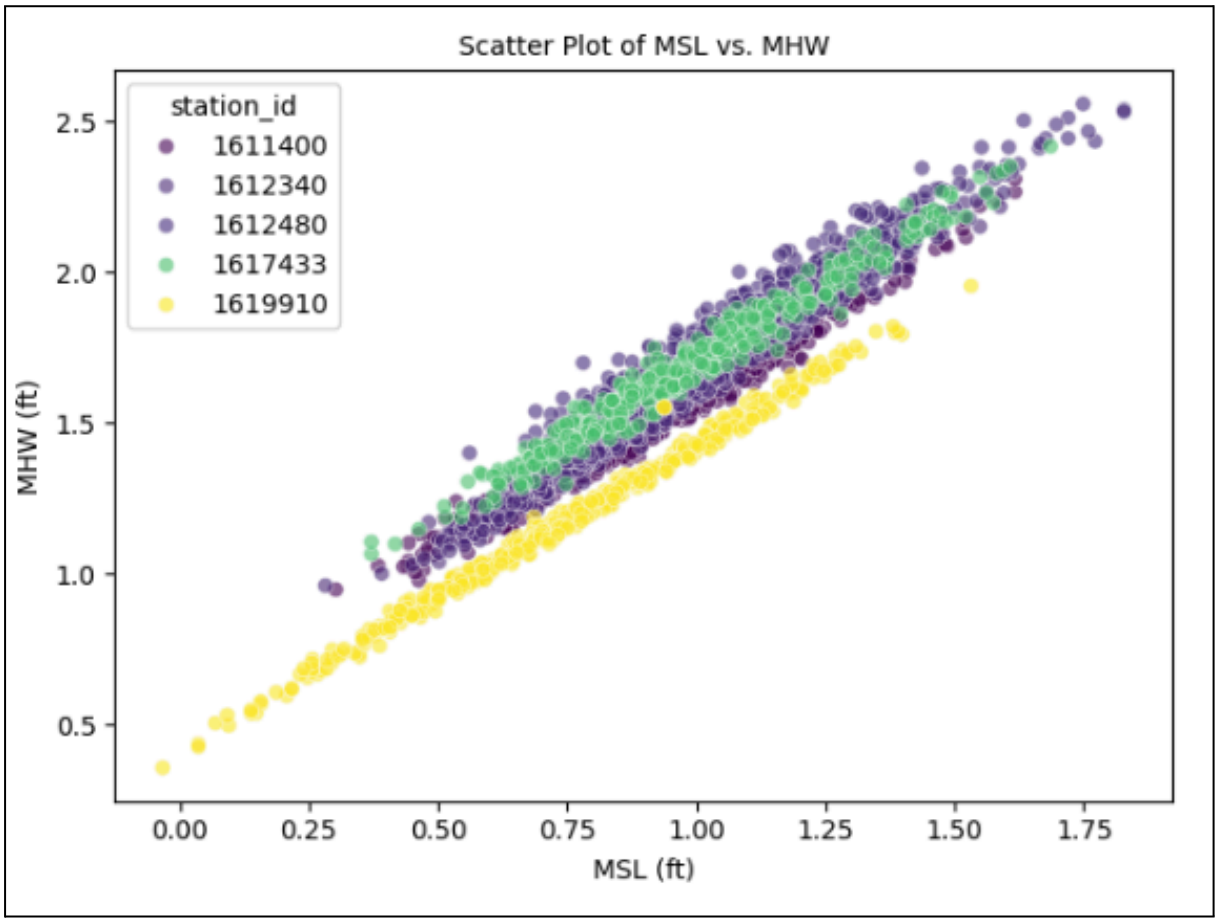
Boxplot of water levels by station:



Insights:

The boxplot graph displayed above compares the highest water levels for five stations - 1611400, 1612340, 1612480, 1617433, 1619910. It represents the variations in medians, interquartile ranges, and outliers in the dataset. Each station shows distinct central tendency values, with station 1619910 having the highest variability and numerous outliers. This suggests that the values observed for station 1619910 have significant fluctuations. On the other hand, stations 1611400, 1612340, and 1612480 display relatively similar ranges and median values, whereas station 1617433 has a slightly elevated median value but fewer outliers. In this way, this visualization emphasizes the difference in water level behaviors among different stations, and indicates that there are some location-specific factors that might be influencing the water levels.

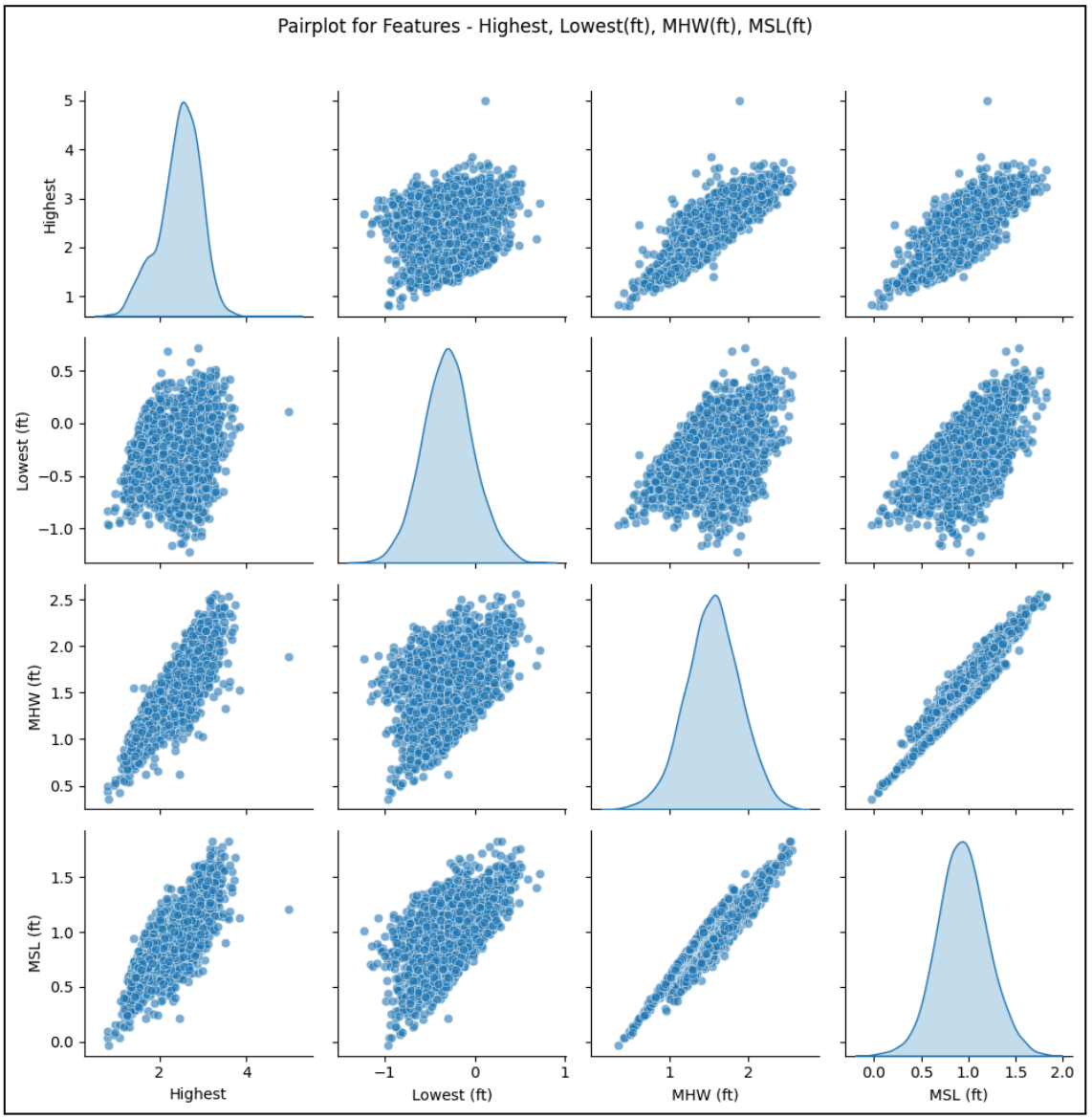
Scatter plot of MSL vs. MHW:



Insights:

The scatter plot of MSL (Mean Sea Level) vs. MHW (Mean High Water) highlights a strong positive linear relationship, and indicates that as MSL value increases, MHW value rises proportionally across all the 5 stations. We have represented different station-specific clusters using different colors to understand the station-specific relationship. Like the station 1619910 shows lower ranges for both metrics and other stations like 1617433 exhibit higher values. Additionally some stations, like 1611400 and 1612340, show overlapping patterns, and the subtle variations represent distinct tidal behaviors. The strong correlation represented using scatter plot underscores the importance of both MSL and MHW as critical features for predicting tidal heights.

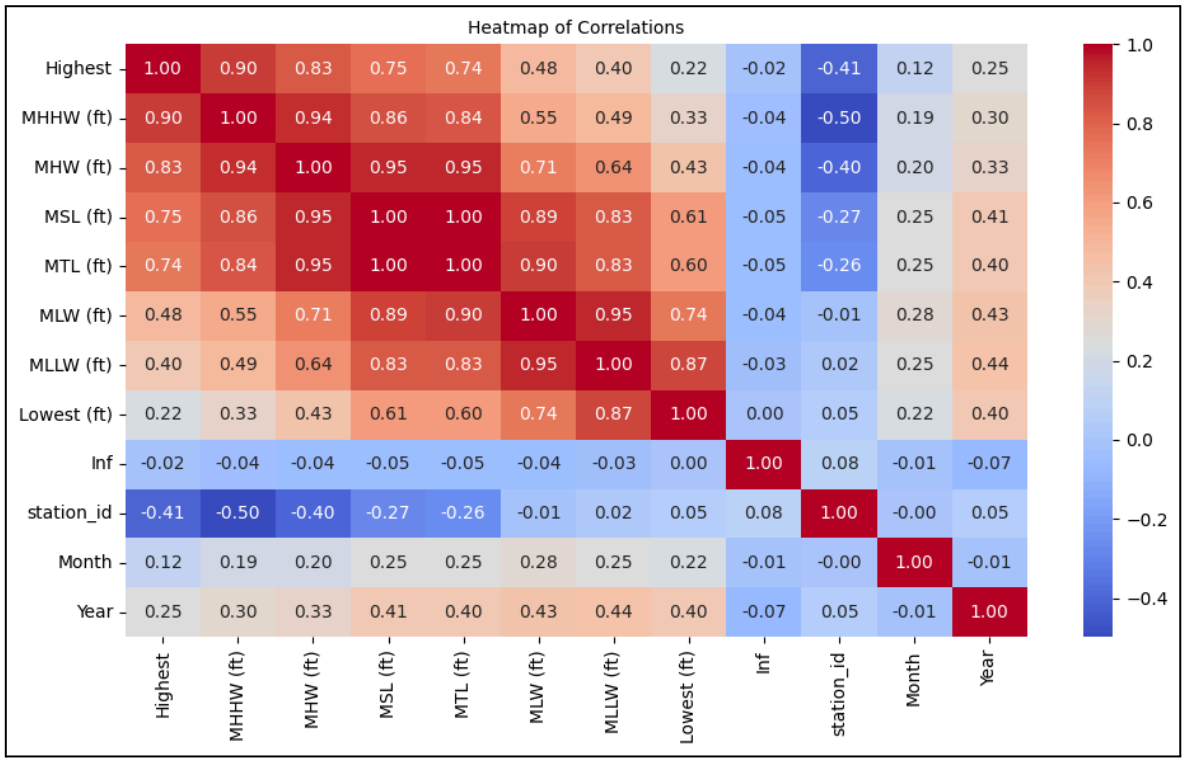
Pairplot for selected features:



Insights:

We have used the pairplot for providing a comprehensive view of relationships between different features like Highest, Lowest (ft), MHW (Mean High Water), and MSL (Mean Sea Level). The diagonal plots illustrate the distribution of each feature, and highlight the skewed behavior of Highest and MHW features and also show the normal distribution of the Lowest and MSL variables. The off-diagonal scatter plots represent the strong positive correlation between MSL and MHW features, and between the Highest and MHW features, and indicate that these features are highly interdependent. On the other hand, the relationship between Lowest and other features is less pronounced, and suggests a weaker contribution to the target variable. Basically, this visualization highlights the critical features in the dataset and drives tidal predictions to suggest which variables are most relevant in order to consider in developing machine learning models.

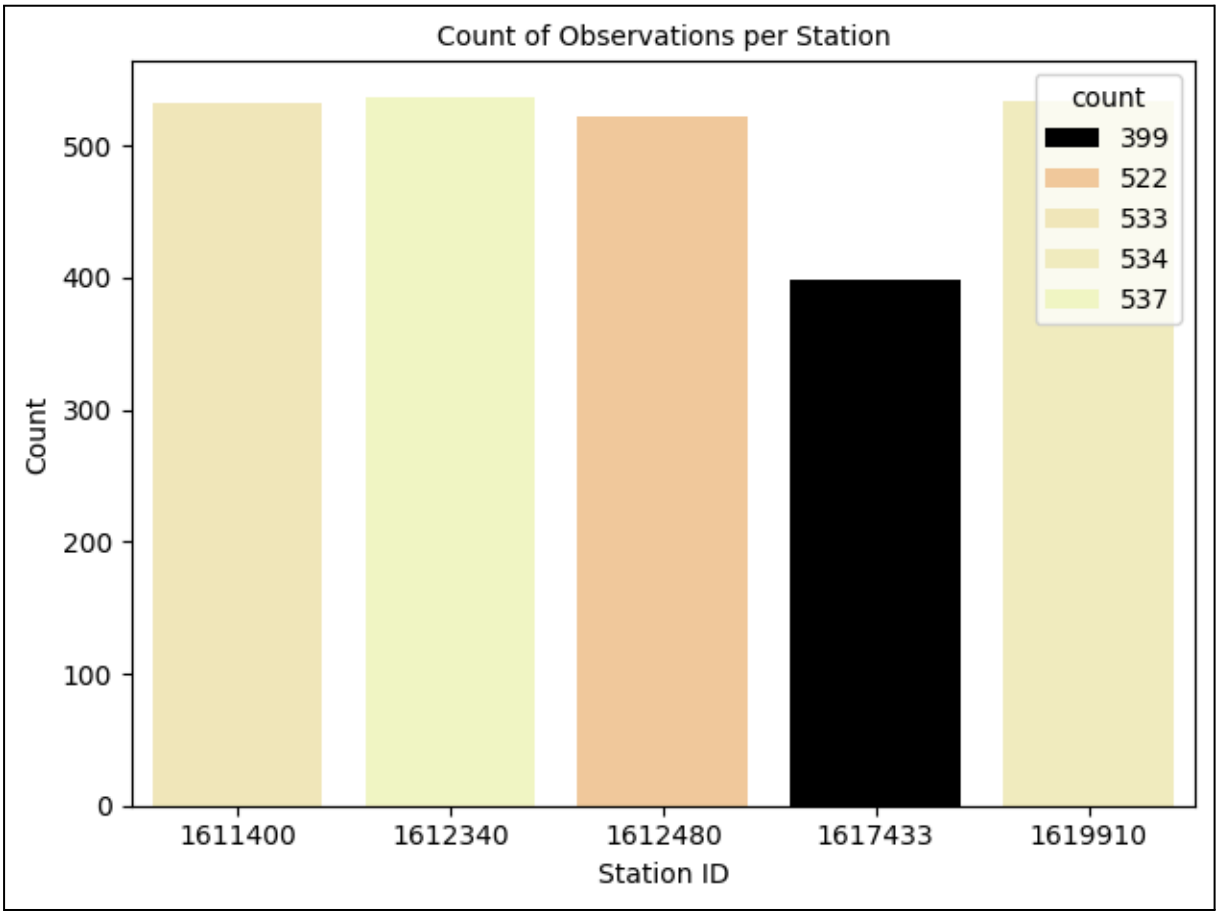
Heatmap of correlation matrix:



Insights:

The heatmap of the correlation matrix above focuses on representing the relationships among the different features in the dataset. The correlation values range from -1 (strong negative correlation) to +1 (strong positive correlation). "**Highest**" feature shows the strongest positive correlations with **MHHW (ft) (0.90)**, **MHW (ft) (0.83)**, and **MSL (ft) (0.75)**, thereby indicating that these features are significant predictors for tidal heights. Similarly, the attributes **MHW (ft)** and **MSL (ft)** are highly correlated with each other **(0.95)**, and reflect their interdependence in tidal dynamics. Variables like **Lowest (ft)** and **MLLW (ft)** represent weaker correlations with the **Highest** (0.22 and 0.40, respectively) attribute, and hence suggest that they contribute less to the target. Station-specific features like **station_id** exhibit moderate negative correlations with feature **Highest**. Additionally, the temporal features like **Year (0.25)** and **Month (0.12)** represent relatively weak relationships. In this way, this heatmap plot provides us with valuable insights for selecting features, and highlights the importance of measuring tidal metrics for predicting tidal heights.

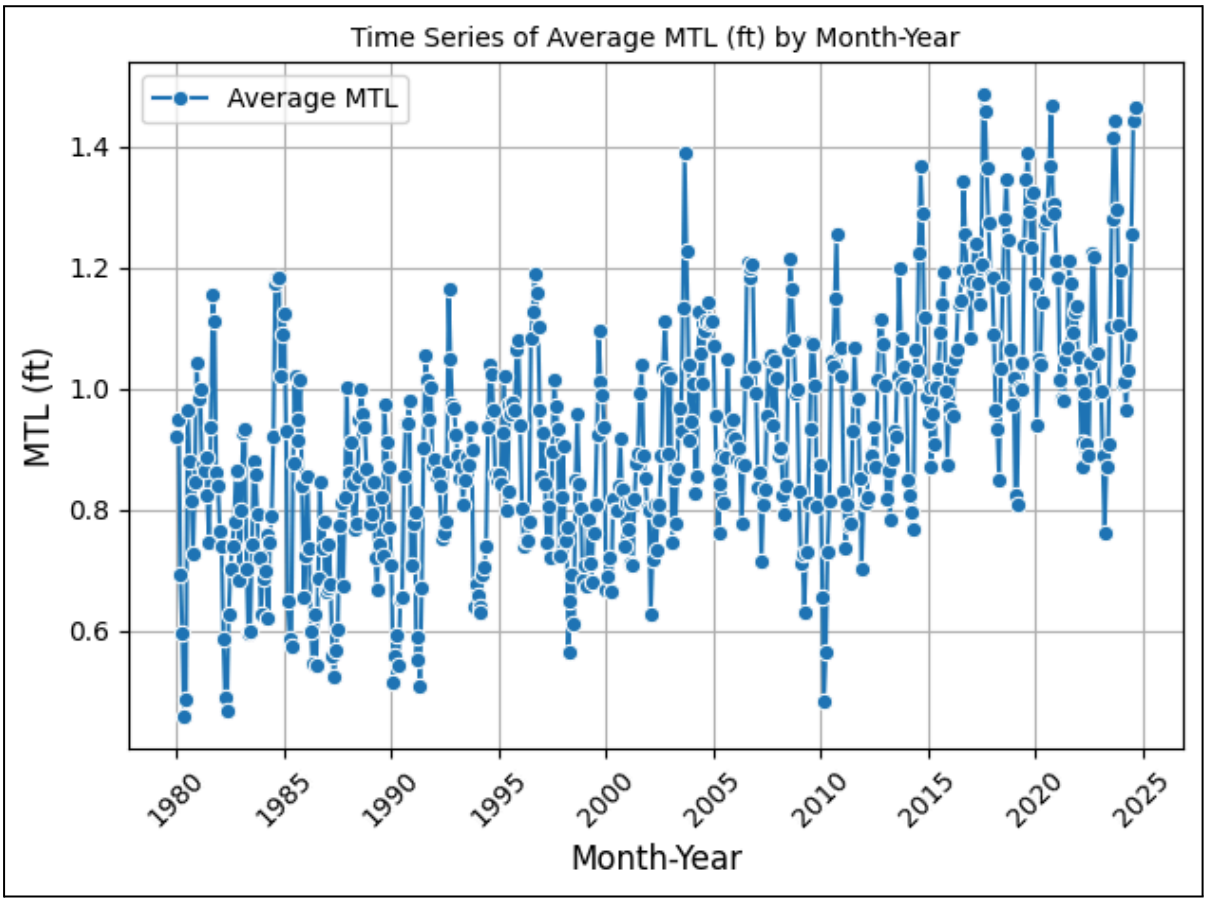
Bar chart for the count of records per station:



Insights:

The bar chart represents the count of records observed for each station in order to illustrate the distribution of available records across the dataset. Based on the above visualization we can observe that the stations 1611400, 1612340, 1612480, and 1619910 each have similar record counts, ranging between 522 and 537. However, the station 1617433 has significantly fewer records like 399 records. This disparity and difference in the record count indicate slight imbalance in the dataset and might affect the model's ability to generalize well for the station 1617433, because fewer observations will provide less information for learning station-specific patterns. Basically this chart helps us in emphasizing the importance of balancing the data while developing and training the models.

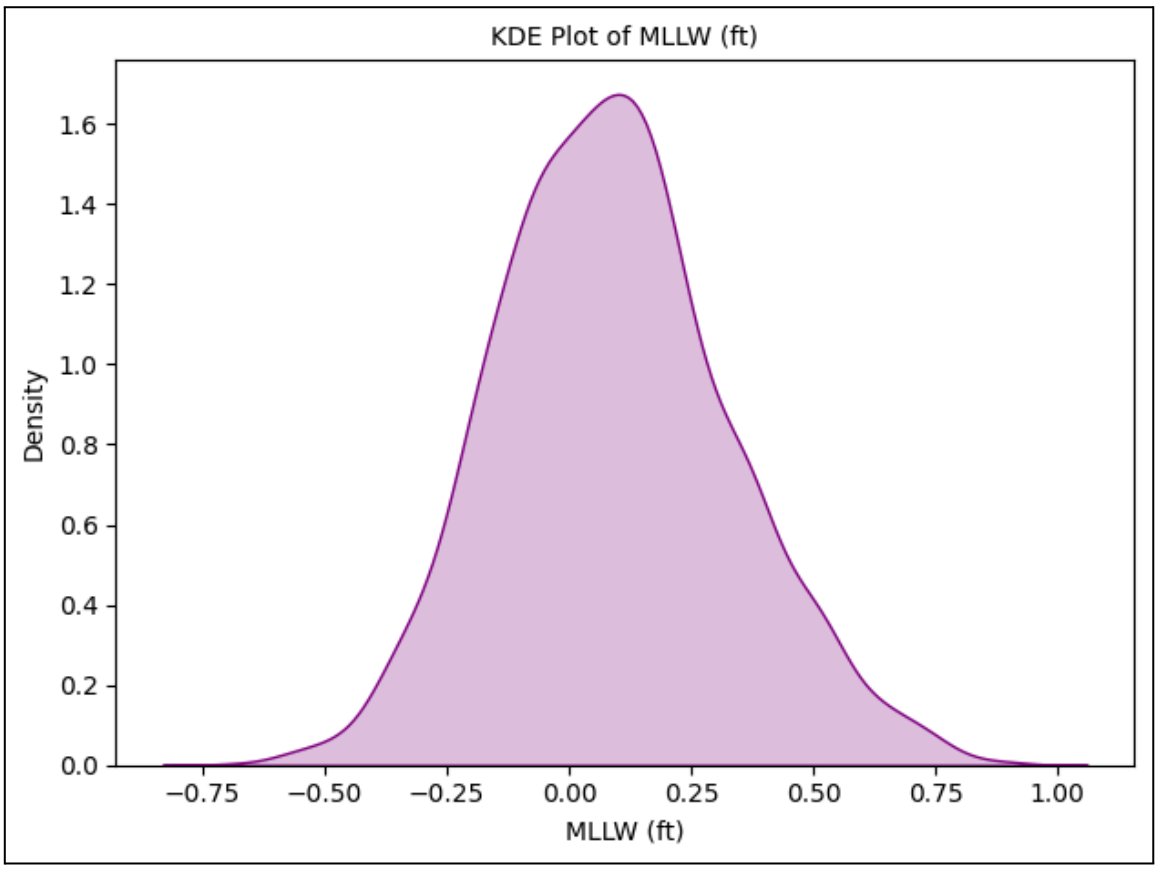
Line plot of MTL (ft) across time:



Insights:

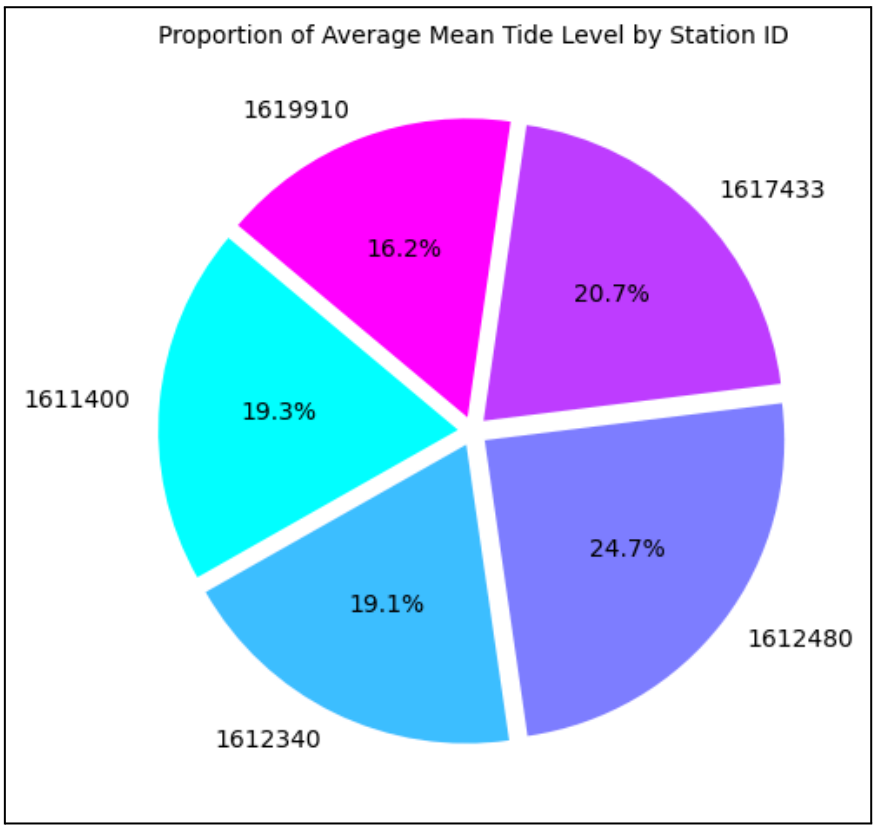
The time series plot of Mean Tide Level (MTL) represents the trends and patterns in tidal behavior observed over time. It represents the gradual upward trend from 1980 to 2025, and highlights a steady increase in MTL (ft) over the past decades. This suggests that possible long-term environmental changes definitely make an impact on tidal behavior leading to rising sea levels. From this visualization we can observe more dynamic tidal activity particularly after 2010. Therefore, this visualization underscores the significance of considering temporal trends when building predictive models for such scenarios like predicting the tidal behavior.

KDE plot of 'MLLW (ft)':



Insights:
The Kernel Density Estimation (KDE) plot of MLLW (Mean Lower Low Water) displayed here represents the distribution of this tidal measure across the dataset for different stations. We can observe that the distribution is unimodal, and has a peak around 0 ft. This indicates that most of the MLLW values are concentrated near this central value. It can also be observed that the density decreases symmetrically on either side of the peak and we can notice a relatively normal distribution with a slight right skew. This implies that higher MLLW values are slightly more common than lower ones, but extreme values in either direction are rarely observed. So, in this way, the KDE plot provides us valuable insights into the central tendency and spread of MLLW and helps to identify its variability so that we can use it as a feature in predictive models accordingly.

PieChart of Average MTL (Mean Tide Level) by Station ID:



Insights:
The pie chart here illustrates the proportion of the average Mean Tide Level (MTL) observed by each station in the dataset. We can observe that station 1612480 contributes the largest share of 24.7% of the total MTL, whereas station 1619910 contributes the smallest share at 16.2%. Additionally, stations 1611400, 1612340, and 1617433 contribute relatively balanced proportions, and range between 19.1% and 20.7%. This distribution reflects variations in tide behavior across different stations, where it can be observed that some stations experience higher average tide levels as compared to others. The pie chart visualization effectively highlights the station-specific differences in MTL values. This becomes essential for understanding and modeling tidal dynamics at different locations.

Conclusion:
In this way, all these visual representations not only supported the interpretation of the dataset but also highlighted the crucial insights that might have been ignored in raw data. Overall, we believe that these visualizations served as a bridge between the complex dataset for different stations and the actionable insights derived from it. We believe that this data analysis will make it easier for us to grasp the insights and utilize them for informed decision-making and even for developing predictive models.

MODEL IMPLEMENTATION:

Decision tree and random forest

Dataset Description

The dataset basically includes different features derived from environmental and tidal measurements across five stations. The target variable we have considered here is the highest tidal measurement (Highest), and the predictors include mean tidal levels (e.g., MHHW, MHW, MLW), categorical station id values, and extracted temporal features like Year, Month, Day, Hour. We handled missing data using mean imputation technique for numerical features, and we encoded the station_id variable as the categorical feature.

Objective

Our primary objective behind this analysis was to develop regression models that would help us predict the highest tidal level (Highest) using both a **Decision Tree Regressor** and a **Random Forest Regressor**. We have also compared the two models based on performance metrics and interpretability.

Below is a detailed description of the model and its evaluation based on the provided code, outputs, and visualizations:

Modeling Process

1. **Decision Tree Regressor**

- We trained the Decision Tree model using default parameters and later fine-tuned through hyperparameter tuning using GridSearchCV.
- Best parameters:
    - max_depth: 5
    - min_samples_leaf: 5
    - min_samples_split: 2
- The optimized model provided an **MSE** of 0.040996 and an **R^2 score** of 0.842 on the test dataset.

2. **Random Forest Regressor**

- A Random Forest model was also trained using the ensemble's ability to reduce overfitting.
- The model was fine-tuned using GridSearchCV. Optimal hyperparameters included:
    - max_depth: 10
    - n_estimators: 200
    - min_samples_leaf: 1
    - min_samples_split: 2
- The optimized Random Forest outperformed the Decision Tree, yielding an **MSE** of 0.033556 and an **R^2 score** of 0.871.

Feature Engineering

1. **Temporal Features:**
    - We converted the Date column into meaningful temporal attributes such as:
        - **Year**, **Month**, **Day** (extracted from the date).
        - **Hour** (derived from the time field).
    - These transformations added temporal granularity to the dataset, and helped us capture seasonal or time-of-day effects on tidal levels.
2. **Categorical Encoding:**
    - We converted the station_id categorical variable to one-hot encoded features. This transformation allowed the model to effectively use information about each station without imposing any ordinal assumptions.
3. **Handling Missing Values:**
    - We replaced the missing values in numerical columns with the column mean using a **Simple Imputer**, and this ensured that the dataset was clean and usable for modeling.

Hyperparameter Tuning

Purpose:
We conducted hyperparameter tuning to optimize the performance of the models for both the Decision Tree and Random Forest regressors using **GridSearchCV** with 3-fold cross-validation.

1. **Decision Tree:**
    - **Parameter Grid:**
        - max_depth: [5, 10, 15, None]
        - min_samples_split: [2, 5, 10]
        - min_samples_leaf: [1, 2, 5]
    - **Best Parameters Identified:**
        - max_depth = 5
        - min_samples_split = 2
        - min_samples_leaf = 5
    - **Performance After Tuning:**
        - MSE: 0.040996
        - R^2: 0.842
2. **Random Forest:**
    - **Parameter Grid:**
        - n_estimators: [50, 100, 200]
        - max_depth: [5, 10, 15, None]
        - min_samples_split: [2, 5, 10]
        - min_samples_leaf: [1, 2, 5]
    - **Best Parameters Identified:**
        - max_depth = 10
        - n_estimators = 200
        - min_samples_split = 2
        - min_samples_leaf = 1
    - **Performance After Tuning:**

    - MSE: 0.033556
    - R^2: 0.871

Insights on Hyperparameter Tuning

- **Decision Tree:**
    - By limiting the depth of the tree (max_depth = 5), the model avoided overfitting and improved its generalization to new and unseen data.
    - Increasing min_samples_leaf to 5 helped us prevent splits with very few data points, further reducing data variance.
- **Random Forest:**
    - Increasing the number of estimators (n_estimators = 200) allowed the model to average predictions across more trees, reducing variance and improving accuracy.
    - Setting a maximum tree depth (max_depth = 10) prevented overfitting while maintaining sufficient complexity to model the relationships.
    - Fine-tuning min_samples_leaf and min_samples_split helped us ensure the balanced splits and enhanced generalization.

Model Evaluation

Actual vs. Predicted Values

- We demonstrated the strong alignment between actual and predicted values using scatterplots.
- We observed that the Random Forest model showed tighter clustering around the perfect prediction line as compared to the Decision Tree model. This helped us with achieving better generalization and reduced prediction error.

Residual Analysis

- We demonstrated the histogram of residuals and it indicated that the Random Forest model has a more symmetric and centered error distribution. This implies better accuracy and less bias as compared to the Decision Tree.
- Additionally, we observed no significant patterns between the residuals and.predicted values. As per our knowledge and understanding, this indicates that neither model suffers from heteroscedasticity or systematic bias.

Feature Importance (Random Forest)

The Random Forest model provided us with the insights into the most important predictors:

- Based on our observations, MHHW (ft) was the most influential feature by a significant margin. It was followed by Month, MLLW (ft), and Lowest (ft) attributes. This highlights the importance of mean high water and temporal feature values in predicting the highest tidal measurement.
- Other features such as Year and MLW (ft) had comparatively lower importance.

Comparison of Model Performance:

Mean Squared Error (MSE)

- Random Forest: 0.033556
- Decision Tree: 0.040996

The Random Forest model demonstrated lower MSE, indicating better accuracy.

R^2 Score

- Random Forest: 0.871
- Decision Tree: 0.842

The Random Forest model achieved a higher R^2 score, reflecting its superior ability to explain the variance in the target variable.

Conclusion

- Therefore, in this way we can conclude that the **Random Forest Regressor** outperformed the **Decision Tree Regressor** in all evaluation metrics, including MSE and R^2. The ensemble approach of Random Forest effectively reduced overfitting and captured complex relationships within the data.
- Feature importance analysis helped us reveal that tidal features (e.g., MHHW) and temporal attributes (e.g., Month) were critical for making accurate predictions.
- We also observed that while the Decision Tree model provides simplicity and interpretability, the Random Forest model offered us significantly better accuracy and robustness.

XGBoost +Prophet

Objective
This analysis uses both time series modeling and machine learning to create a better predictive model to forecast the highest tidal levels, Highest. Tidal predictions are very essential in various applications such as coastal management, flood prevention, navigation, and environmental monitoring. While classical time series models like Facebook Prophet are really good at capturing seasonal patterns and long-term trends, they often fail when it comes to complex nonlinear relationships in the data. On the opposite side, machine learning models like XGBoost will be able to model these relationships but will not account intrinsically for temporal dependencies as well as time series models do.

Dataset Preprocessing

- **Feature Extraction:**
    - The temporal features Year, Month, Day, Hour were extracted from columns for Date and Time GMT, which will help capture seasonal and temporal patterns of the data.
    - The 'ds' column was formed by joining the 'Date' and 'Time' for compatibility with the Prophet model.
- **Missing Data Handling:**

- Missing values for numerical columns were imputed with the mean using a SimpleImputer.
- **Categorical Encoding:**
  - The column station_id was converted into categorical features through one-hot encoding for the XGBoost model.

Step 1: Time Series Forecasting with Facebook Prophet

- The Prophet model was trained with the highest tidal measurement as the target variable, and the ds column as the time variable.
- **Model Assumptions:**
  - Seasonalities (e.g., yearly) are automatically inferred by the Prophet.
  - Daily and weekly seasonalities were disabled due to the nature of the data.
- **Predictions:**
  - The model provided yhat values (predictions) for Highest.
  - Residuals (residuals = Actual - yhat) were calculated to quantify errors from the initial model.

Performance:

- Mean Squared Error (MSE): High (as indicated in the comparison plots).
- R^2 Score: Relatively low due to the inability to capture complex patterns in the data.

Step 2: Residual Correction using XGBoost

- Purpose:
  The residual errors of Prophet were modeled by a powerful tree-based regression model called XGBoost, which would model additional patterns that the time series model missed.
- Feature engineering:
  Selected numerical tidal features (MHHW (ft), MHW (ft), etc.), temporal features (Year, Month, etc.) and one-hot encoded station_id variables are exposed as predictors to the model.
- Model tuning:
  A simple model of XGBoost was trained first, giving the following performance:
  MSE: 0.0304
  R^2: 0.8536
  Used GridSearchCV to fine tune hyperparameters:
- Best parameters:
  max_depth = 3
  learning_rate = 0.1
  n_estimators = 200
  subsample = 0.6
  colsample_bytree = 0.6
- Performance after tuning:
  MSE: 0.0296
  R^2: 0.8572
- Improvements were seen on MSE and R^2 due to reduced overfitting as well as better parameter tuning.

Step 3: Hybrid Model

- **Combination:**
    - Final predictions were obtained by adding the Prophet predictions (yhat) with the corrected residuals predicted by XGBoost as follows:
    - Final Prediction = yhat (Prophet Prediction) + XGBoost Residual Prediction
- **Results:**
    - The hybrid model greatly improved accuracy, beating both individual models.

The objective is twofold:

Develop an Initial Forecast with Prophet:

- Used Facebook Prophet to capture temporal patterns in this data set, such as yearly seasonality and long-term trends.
- Made a baseline prediction for the Highest tidal levels (yhat) so a preliminary understanding of the tidal dynamics can be established.

Enhance the Forecast by Correcting Residual Errors:

- Here we have identified and modeled the residuals/errors in the Prophet predictions since these represent the unexplained gaps in its power.
- It should use the powerful regression model, XGBoost, in analyzing the residuals for correcting these complex nonlinear relationships that are not captured by Prophet.
- Also combined the Prophet predictions and XGBoost corrections to produce a final, more accurate prediction.

Evaluate the Hybrid Model:

The performance of this hybrid model was then evaluated against Prophet and XGBoost models. This analysis showed that the hybrid approach outperforms both methods in terms of reducing MSE and strengthening explanatory power, as reflected by a higher R^2 score. This strategic combination utilized Prophet to learn from seasonal trends, temporal dependencies, and the overall patterns, while XGBoost served as the corrective layer to handle shortcomings of Prophet's forecasts, especially about outliers or deviations produced by nonlinear relations. The ultimate goal was to develop a strong, interpretable predictive model that combines the strengths of time series analysis and machine learning into a highly performing system for tidal level forecasting.

Significance of Hyperparameter Tuning to Reach the Goal

The most important role of hyperparameter tuning was in significantly improving the results of the hybrid model. This was because, with Prophet, it tuned the baseline forecast to capture the underlying structure of the data by aligning the model's assumptions-for instance, yearly seasonality and flexibility around changepoints-with the actual characteristics of the data. Similarly, XGBoost also uses a systematic parameter optimization

to enable the model to correct residual errors by capturing intricate patterns and nonlinear relationships from the data.

The objective of reducing forecasting errors and improving predictive accuracy was achieved by the careful tuning of both components. This hybrid approach emphasized how a well-tuned combination of time series modeling and machine learning might lead to superior performances compared to standalone models, thus demonstrating a potential for high performance in tidal level forecasting.

Model Comparison

The performances of the hybrid model were compared against the Prophet and XGBoost models with standalone uses in terms of key metrics, such as Mean Squared Error (MSE) and R^2 score, to decide its efficacy in tidal level forecasting.

Mean Squared Error (MSE):

Prophet Only: Had a high error because it lacks the capability to catch intricate, nonlinear relationships, which naturally exist in tidal data.

XGBoost on Residuals: It decreased the error by a big margin because it can effectively model the residual, which was left by Prophet while predicting.

Hybrid Model: It has shown the lowest MSE because this model combines Prophet's abilities to handle trend and seasonality with XGBoost for refining residuals.

R^2 Score:

Prophet Only: Had low explanatory power; this points to its inability to fully model the complexities of the data.

XGBoost on Residuals: A big improvement, with a high R^2 value of 0.8572, which points to its good alignment with actual values.

Hybrid Model: Slightly outperformed XGBoost alone with the highest R^2 score, hence robust predictive performance coupled with good explanatory power.

Insights:

• MSE Comparison - Bar Chart: Demonstrated the large decrease in error from a model using Prophet to XGBoost and finally to the hybrid model.

• Comparison of R^2 - Bar Chart: Depicted the hybrid model to be more effective by far in explanation when compared to single models.

This analysis underlines how a hybrid model can perform better compared to univariate approaches with the complementarity of time series analysis and machine learning.

Conclusion

The hybrid model successfully combined the time series capabilities of Prophet with the residual correction strength of XGBoost, thus achieving a superior performance in predicting tidal levels.

Strengths of Hybrid Approach:

- **Comprehensive Analysis:** Merged the strengths of Prophet in capturing temporal patterns, including seasonality and trend, with XGBoost's strength in modeling potentially complex nonlinear relationships in residuals.

• **Improved Accuracy:** The hybrid model decreased the error of prediction and led to better overall performance by incorporating the two approaches.

• **Flexibility and Modularity:** The residual correction step is adaptable, whereby any advanced machine learning model, such as XGBoost, can be replaced, thus making the approach versatile for different datasets and use cases.

LSTM+Dense Model implementation

Objective

The main objective of this work is to forecast the "Highest" tide level with an LSTM-based neural network. Considering the nature of the features, which are time-dependent, LSTMs will be suitable for this problem. This project will preprocess the data by feature engineering and optimize the model architecture in a way that produces efficient and reliable results with minimal errors.

Step 1: Data Preprocessing

The dataset needed effective preprocessing to make it ready for the LSTM model. This included handling missing values, engineering features, encoding categorical variables, scaling numerical data, and reshaping input data for time-series modeling.

1. Handling Missing Values:

- Numeric Features: For numerical columns, missing values were replaced by the mean to make the values consistent without introducing bias.

- Non-Numeric Features: Missing values in the categorical columns were filled with the mode. It was also necessary to ensure that this step made logical sense.

2. Temporal Feature Engineering:

Date and Time Extraction: The "Date" column was converted to datetime, and further features such as "Month," "Day," and "Hour" were extracted from it to capture the temporal patterns.

- Cyclic Time Encoding: To represent time as cyclic, sine and cosine transformations were applied to the "Hour" feature:

- `Sin_Hour = sin(2π * Hour / 24)`

- `Cos_Hour = cos(2π * Hour / 24)`

This ensured that the model learned the proximity of a cycle between start and end in a 24-hour setting.

3. Categorical Encoding:

The column station_id was one-hot encoded using OneHotEncoder to create separate binary features for each station. In this way, it will not introduce any ordinal bias, and the model will treat each station as an independent entity.

4. Feature Scaling:

A MinMaxScaler was applied to numerical features like "MHHW (ft)," "MHW (ft)," etc., normalizing their values within the range of 0 to 1. This step was essential for ensuring efficient training of the LSTM model, which is sensitive to feature scales.

5. Train-Test Split:

The dataset was divided into 80% training and 20% testing using `train_test_split`.

Input data (`X`) was then reshaped into a 3D format with dimensions as shown: [samples, timesteps, features]`. For instance, in this case, the shape for training data is (2020, 1, 18), while for testing data, it is (505, 1, 18).

Step 2: Model Development

An LSTM-based neural network model developed with a notion for capturing the sequential nature of the data that refers to tides. Care has been taken in the architecture, compilation, and optimization steps in order to balance complexity, accuracy, and computational efficiency.

2.1 LSTM Architecture

1. Layers
-**LSTM Layer:**
Units: The layer was composed of 64 neurons, each with the tanh activation function for the effective learning of sequential dependencies and temporal patterns within the data.
- **Dropout Layer:**
Dropout Rate: A dropout rate of 20% was done to avoid overfitting by randomly deactivating neurons during training; this regularization technique improves the generalizing capability of the model.
-**Dense Layer:**

Units: The dense layer consisted of 32 neurons with ReLU (Rectified Linear Unit) activation to learn and model complicated nonlinear relationships in the data.

-Output Layer:

It contained only one neuron with a linear activation function to provide continuous outputs for regression tasks.

The Purpose is to directly predict the target variable, the highest tidal level ("Highest").

2. Compilation

Optimizer: The Adam optimizer was used due to its adaptive learning nature, with a learning rate of 0.001, which enabled effective convergence during training.

Loss Function: MSE was employed to minimize the squared differences between predicted and actual values, as larger errors are considered significant.

Metrics:MAE was used to check the average prediction error of a model. It yielded an interpretable measure of accuracy.

Step 3: Training the Model

3.1 Training Procedure

The model is trained on 50 epochs with a batch size of 32 for efficient learning to converge. A validation set of 20% of the overall dataset is used to keep tabs on the performance during training.

- Epochs:50

- Batch Size:32

- Data for Validation: The validation set consisted of `(X_test, y_test)` and was used to compute metrics at the end of each epoch.

Output:

The training and validation losses with respect to the Mean Squared Error-MSE and Mean Absolute Error-MAE, respectively, were computed for both training and validation datasets in order to keep track of the model's progress.

3.2 Training Results

The training process was characterized by the steady and smooth convergence of both the training and validation loss curves, hence confirming that the model generalizes without overfitting. Therefore, the final validation loss can be regarded as MSE of 0.0373, while the final validation MAE is 0.1377.

Step 4: Model Evaluation

4.1 Metrics

The model performance on the test dataset is as follows:

Mean Squared Error (MSE): 0.0373

- MAE: 0.1377

These metrics reflect the fact that the LSTM model greatly minimized the errors of prediction and provided reliable forecasts of tidal levels.

4.2 Insights:

1. Training and Validation Loss:

- A comparison plot of training vs. validation loss over 50 epochs was characterized by smooth convergence; the validation loss was following closely with training loss, reflecting successful learning without huge overfitting.

2. Actual vs Predicted Values:

- A scatterplot of actual vs predicted values showed a strong alignment along the diagonal, indicating a high degree of predictive accuracy.

3. Residual Analysis:

- A histogram of residuals (errors) was symmetric and narrow around zero, indicating unbiased predictions with minimal systematic error.

Key Observations

1. Model Strengths:

- The LSTM architecture substantially captured the sequential patterns in tidal data, producing low MSE and MAE.

- Temporal feature engineering, such as sine and cosine transformations, enhanced the cyclic time interpretation capability of the model.

2. Limitations:

- Even with an overall strong performance, the model clearly struggled with extreme outliers, as shown by slight scatter in the scatterplot of predictions.

- Further tuning could be done to enhance robustness and performance of the models by tuning the number of LSTM units or changing the dropout rates.

Conclusion

The LSTM model showed very good potential for tidal level prediction by confirming its great accuracy. By using its strength in capturing sequential dependencies and partnering it with elaborate data preprocessing and a well-structured architecture, the model ensured low error metrics and robust predictive performance. Future enhancements could focus on advanced hyperparameter tuning, exploring additional temporal and spatial features, or incorporating ensemble approaches to further refine and improve the accuracy of predictions.

Model comparison table:

Model	MSE	R^2 Score	Strengths	Limitations	Best Use Case
Decision Tree Regressor	0.0409	0.842	Simple, interpretable, and fast. Captures basic relationships in the data.	Susceptible to overfitting, limited in handling nonlinear relationships.	Quick, interpretable models with small datasets.
Random Forest Regressor	0.0336	0.870	Reduces overfitting using ensemble learning. Handles feature importance analysis effectively.	Requires more computational power; less interpretable than Decision Tree.	Medium to large datasets with complex patterns.
Prophet Only	~0.175	~0.20	Excellent at capturing time series trends and seasonality.	Struggles with nonlinear relationships and outliers.	Time series forecasting with periodic patterns.
Prophet + XGBoost (Hybrid)	0.0296	0.857	Combines time series analysis with machine learning for better accuracy. Handles nonlinear residuals.	More complex to train and tune; dependent on both models' performance.	Hybrid tasks with sequential and nonlinear data.
LSTM Neural Network	0.0373	0.8431	Captures sequential patterns and dependencies in time series data. Good at modeling relationships over time.	Requires significant computational resources and careful preprocessing. Sensitive to overfitting.	Sequential data with time dependencies.

Detailed Insights and Observations

The **Decision Tree Regressor** performed great with an **MSE of 0.0409 and an R^2 score of 0.842.** Despite its simplicity and interpretability, the Decision Trees tend to overfit in cases of big datasets and complex patterns in them. Due to this overfitting issue, it is not that effective for generalizing the predictions on new, unseen data. Decision Trees should be chosen when model transparency and ease of interpretation are needed or if there is a need for computational efficiency.

In contrast, the **Random Forest Regressor** had a much better performance: **its MSE was equal to 0.0336 and the R^2 score equaled 0.870.** By averaging the predictions of multiple Decision Trees, Random Forest overcomes the overfitting problem and is able to model more complex patterns within the data. Actually, this kind of ensemble not only improved the predictive accuracy but also enhanced the robustness of the model. While random forests are suitable for problems where high accuracy is desired along with feature importance assessment, they are especially useful in those scenarios where both performance and interpretability should go alongside.

The **Prophet model**, with an approximate **MSE of 0.175 and an R^2 score of 0.20**, demonstrated its limitations when applied to this dataset. While it was effective at capturing general trends and seasonality, Prophet struggled with nonlinear patterns and data variability, which resulted in a relatively low performance. This model is best suited for datasets with clear temporal trends or periodicity but may not perform well when complex nonlinear relationships are present, making it less ideal for datasets with significant variability or outliers.

The **Hybrid Model** that combined Prophet with XGBoost outperformed all other models **with an MSE of 0.0296 and an R^2 of 0.857**. By leveraging Prophet's ability to forecast trends and seasonality, alongside XGBoost's strength in handling residual errors, the hybrid model successfully addressed both sequential dependencies and nonlinear relationships in the data. Despite its complexity and the need for tuning both models, this hybrid approach proved to be a versatile solution for datasets with mixed characteristics. Its superior performance highlights the potential of combining time series forecasting with machine learning for robust and accurate predictions.

The **LSTM Neural Network**, with an **MSE of 0.0373**, proved to be highly effective at capturing sequential patterns and dependencies in time-dependent data. However, the model requires significant computational resources, and proper tuning and regularization are necessary to avoid overfitting. LSTMs are particularly well-suited for datasets with long-term dependencies or when the data exhibits complex, highly sequential patterns. While its performance was competitive, the model's complexity and resource requirements make it a more demanding choice compared to simpler models or hybrids.

Model Suitability for Tidal Predictions based on the results received:

Among these, the **Hybrid Model Prophet + XGBoost** had the lowest MSE and highest explanatory power by combining temporal modeling with Prophet and the residual error-handling capability of XGBoost. This hybrid model effectively captures both the sequential dependencies and nonlinear relationships, making it most suitable for high-accuracy tidal level predictions.

The **Random Forest Regressor** performed second best and had a strong MSE and R^2 score while keeping the model structure simpler and more interpretable compared to the hybrid model. That is an excellent choice whenever interpretability of feature importance is imperative, considering that good performance is obtained with less complexity.

The **LSTM neural network** will be most appropriate in applications whose dataset is predominantly dependent on sequences or provides a complex pattern for prediction and, hence, is suited for problems that involve modeling long-range dependencies. Although computationally costly and prone to overfitting, it provides good outputs in highly sequential and time-dependent data.

TCN and TCN+LSTM

Objective: The objective is to be able to forecast MSL by using state-of-the-art architectures, including:
Designing our own best-performing Temporal Convolutional Network (TCN) model
Using the best-performing, most hybrid TCN-LSTM model by combining the strengths from both.
During this project, additional steps were taken to finetune model performance by hyperparameter tuning, developing regularization, and injection of noise to help avoid overfitting.

Step 1: Data Preparation
Preprocessing of the tidal-related features, including MHHW, MHW, and MTL, included imputing missing values and engineering time-based features to extract seasonal and cyclic patterns. StandardScaler was applied to normalize the data, ensuring better convergence for models.

Step 2: Custom TCN
The TCN model used causal convolutions to ensure temporal ordering and employed residual connections to allow better gradient flow during training.
Dilation rates were applied to capture long-term dependencies.

Initial Results:
MSE = 0.0279
R^2 = 0.5235

Step 3: Hyperparameter Tuning for TCN
Hyperparameters optimized using Keras Tuner included the number of TCN blocks, filters, kernel size, dilation rate, learning rate, and dense layer units.

Best Hyperparameters:

- Number of TCN blocks: 1
- Filters: 128
- Kernel size: 5
- Dilation rate: 2
- Dense units: 64

- Learning rate: 0.01
- Optimized Results:
- MSE = 0.0230
- R^2 = 0.650+

Step 4: Hybrid TCN-LSTM Model

This hybrid model combined TCN's strengths in capturing local temporal patterns with LSTM's strengths in learning long-term sequential dependencies.

Overfitting was avoided by incorporating the following regularization techniques into the models:

- Dropout layers
- L2 regularization
- Noise injection
- Early stopping
- Optimized Results:
- MSE = 0.0042
- MAE = 0.0254
- R^2 = 0.9799

Evaluation and Insights:

Performance:

The performance of the hybrid TCN-LSTM model proved to be outstanding, with the model explaining 97.99% of variance in MSL data based on the R^2 score value of 0.9799. The very low value of MSE (0.0042) showed that the model expressed temporal dependencies or patterns very well.

Overfitting Mitigation:

The techniques used-dropout, L2 regularization, and noise injection-finally paid off, reducing overfitting while early stopping ensured training stopped at an optimal point before overfitting occurred.

Residual Analysis

The residual histogram was around zero, indicating minimal prediction bias, and most errors were small in magnitude and symmetric in distribution.

Insights:

Training and Validation Loss:

There is a smooth decrease in loss over epochs with no significant overfitting.

Actual vs Predicted Values:

A scatter plot showing close alignment between actual and predicted values confirmed strong predictive accuracy.

Residual Distribution:

Symmetric distribution around zero assured that residuals supported model predictions with no bias.

Conclusion:

The Hybrid TCN-LSTM model combines the very best qualities for forecasting the Mean Sea Level: local temporal patterns using the state-of-the-art TCN and long-term dependencies using LSTM. Careful regularization, hyperparameter tuning, and robust preprocessing led to outstanding performance that balanced accuracy and generalization in large measure, making it ideal for predicting MSL in complex, time-dependent data.

Model comparison

Criteria	TCN Model	Hybrid (TCN + LSTM) Model
Objective	Predict Mean Sea Level (MSL)	Predict Mean Sea Level (MSL)
Architecture	Temporal Convolutional Network (TCN)	Combination of TCN and LSTM layers
Mean Squared Error (MSE)	0.0307	0.0042
Mean Absolute Error (MAE)	0.1333	0.0254
R^2 Score	0.6258	0.9799
Training Duration	Relatively Fast	Longer due to complexity
Model Complexity	Moderate	High
Overfitting Risk	Low to Moderate	Moderate,mitigated with regularization.
Strengths	Stable and interpretable for time-series	Captures both temporal dependencies (TCN) and sequential patterns (LSTM)
Limitations	Limited ability to model long-term dependencies	Risk of overfitting due to model complexity

Key Insights

- The Hybrid, TCN-LSTM, outperforms the singular TCN model by a large margin, as supported by the lower MSE and higher R^2 Score.
- The hybrid model is more complex and computationally expensive, although it carries a higher risk of overfitting.
- The TCN model is faster in training and simpler; therefore, the better choice when computational resources are limited or accuracy requirements are not very high.

Seasonal and Temporal Analysis for Water Levels

Objective of Analysis
This analysis of seasonal and temporal patterns in water levels was conducted along with the use of two approaches, for example: Highest.
1. **Seasonal Decomposition:** Trend, seasonality, and residual variations were identified in water levels.
2. **Rolling Statistics:** Long-term trends were tracked using moving averages to compare the behavior of water level across stations.
Thereafter, both approaches worked towards discovering recurring patterns, detecting any time trends, and drawing useful inferences to permit good planning and sound decision-making.

Step 1: Data Preprocessing
Input Data: Historical water levels and information from stations.
Steps Taken:
The missing values for the Highest water levels have been imputed using forward and backward filling to maintain consistency with no gaps in the time series.
Data was resampled to a monthly frequency to capture broader seasonal and long-term trends.
Normalization was not performed in this analysis since it was pattern recognition and visualization, not model performance-dependent.

Step 2: Seasonal Decomposition
Here seasonal Decomposition was done based on the Highest water levels for each station using STL: Seasonal-Trend decomposition using Loess. It decomposes the time series data into three components: Trend that describes long-term changes in water levels, Seasonality that pinpoints periodic fluctuations at regular intervals (usually annual cycles), and the Residuals, irregularities or anomalies not taken into account by the trend or seasonality. The decomposition revealed some station-specific trends, with some showing a rising water level over time and others staying roughly the same, which could point to environmental changes, climatic variables, or urbanization. Seasonal patterns were quite clear, probably due to tides or weather, which predictably peak at certain times of the year. Residuals represented random fluctuations that showed anomalies or events not modeled by either the trend or seasonal component, such as extreme weather events or unusual tidal behavior, and further enhanced the understanding of the variability in the water level between different stations.

Step 3: Rolling Statistics
In Step 3 we performed a rolling average on the Highest water levels by applying a 12-month smoothing to dampen short-term variability and highlight longer-term trends. This method

served to accentuate slow changes over time in water levels and helped to distinguish long-term trends from short-term variability and seasonal noise. By comparing the rolling averages between these different stations, striking spatial variations in water level averages reveal important regions with consistently higher or lower water levels. These findings are useful for regional planning and for identifying localized water level behaviors. The rolling averages still captured the seasonal peaks and troughs; however, these were less pronounced in comparison with the seasonal decomposition. This would indicate that the rolling averages smoothed out the larger, longer-term trends while dampening most of the short-term seasonal variation related to tidal cycles and weather patterns.

Conclusion

Applying Seasonal Decomposition and Rolling Statistics, we were able to come up with the seasonal and temporal dynamics of water levels across stations. Seasonal decomposition brought out the long-term trend and cyclic seasonal patterns, while rolling statistics helped to get a better view of changes in water level over the long term and variations at the station level. These will hopefully inform future decision-making and planning, especially in the areas of flood risk management, water resource allocation, and infrastructure development.

Evaluation and Insights

Criteria	Seasonal Decomposition	Rolling Statistics
Focus	Breaks data into trend, seasonality, residuals.	Highlights long-term trends using smoothing.
Seasonal Insights	Explicitly identifies recurring patterns.	Seasonal effects are visible but implicit.
Trend Analysis	Clearly separates long-term trends.	Shows trends but mixes them with seasonality.
Anomaly Detection	Easier to spot deviations in residuals.	Requires visual identification of anomalies.

Complexity	More detailed but requires decomposition tools.	Simpler and focuses only on averages.

Key Insights:

Seasonal decomposition brings out the cyclical nature of variation in water level to show which months are most prone to have highest or lowest levels. This assists in making good forecasts of when extreme water levels could happen.
Both seasonal decomposition and the rolling statistics allow for the detection of trends in water levels over time. However, seasonal decomposition isolates these trends explicitly from the other fluctuations, yielding a sharper picture of the longer-term changes.
It allows direct comparisons between stations by using rolling averages. It shows whether water levels are higher in certain stations regularly or more steadier over time, providing insights that can be used for regional planning and managing resources accordingly.

SUMMARY:

To summarize our project work, we used a dataset that consisted of the information on sea-levels from 1980 to 2025 for different sea stations. Our major focus was to predict the tidal levels and analyze sea-level variations using advanced machine learning and deep learning models. By following a series of comprehensive tasks including data preprocessing, data visualizations, and predictive modeling, we were successfully able to identify critical patterns and provided actionable insights which will be useful to enhance coastal management, disaster preparedness, and navigation safety. These outcomes will help policymakers and government environmental agencies to mitigate future environmental risks and foster sustainability in vulnerable coastal areas.