* Data Wrangling: This section includes data preparation and cleaning steps.
    * pandas as pd: Imports the Pandas library for data manipulation.
    * numpy as np: Imports the NumPy library for numerical operations.
    * missingno as msno: Imports the Missingno library for visualizing missing data.

* Predictive Modeling: This section involves building and evaluating machine learning models.
    * LinearRegression: Importing the Linear Regression model from scikit-learn.
    * KNeighborsRegressor: Importing the K-Nearest Neighbors Regressor model.
    * RandomForestRegressor: Importing the Random Forest Regressor model.
    * mean_squared_error, r2_score: Importing evaluation metrics.
    * train_test_split: Importing a function to split data into training and testing sets.
    * statsmodels.api as sm: Importing the statsmodels library for statistical modeling.
    * scipy.stats as stats: Importing statistical functions from SciPy.

* Visualization: This section is for data visualization.
    * seaborn as sns: Imports the Seaborn library for data visualization.
    * matplotlib.pyplot as plt: Imports the Matplotlib library for creating plots.
    * sns.set_style("whitegrid"): Sets the Seaborn style for plots to "whitegrid."

* Warnings: Handling warnings in the code.
    * warnings.filterwarnings("ignore"): Silences warnings to prevent them from being displayed during execution.

* Import dataset into Pandas DataFrame:
    * df_raw: This variable is used to store the data read from a CSV file using the pd.read_csv function. The file is located at "../dat/ship_data.csv."
    * n_records: This variable stores the number of records (rows) in the DataFrame df_raw, which is obtained using df_raw.shape[0].

* Check column names and data types:
    * df_raw.info(): This line of code calls the info() method on the DataFrame df_raw. It displays information about the DataFrame, including column names, data types, and the number of non-null entries in each column. This is useful for initial data exploration and understanding the dataset's structure.

* Plot raw data:
    * plt.figure(figsize=(16, 8)): Sets up the figure size for the plot, making it 16 inches wide and 8 inches tall.

* sns.scatterplot:
    * x = 'Speed Through Water (knots)': Specifies the data to be plotted on the x-axis, which is the 'Speed Through Water (knots)' column from the DataFrame df_raw.
    * y = 'Main Engine Fuel Consumption (MT/day)': Specifies the data to be plotted on the y-axis, which is the 'Main Engine Fuel Consumption (MT/day)' column from the DataFrame df_raw.
    * hue = 'Weather Service Apparent Wind Speed (knots)': This parameter assigns different colors to the data points based on the 'Weather Service Apparent Wind Speed (knots)' column from the DataFrame df_raw.
    * data = df_raw: Specifies the DataFrame from which the data should be extracted for plotting.
    *
This code creates a scatter plot with 'Speed Through Water (knots)' on the x-axis, 'Main Engine Fuel Consumption (MT/day)' on the y-axis, and uses color to differentiate data points based on 'Weather Service Apparent Wind Speed (knots)'.

    * Create helper sublists of column names:
    * cols_main: This sublist includes the first four columns (0 to 3) from the DataFrame df_raw. It appears to represent main information columns.
    * cols_draft: This sublist includes columns 4 to 7 from df_raw, which may relate to draft information.
    * cols_shaft: This sublist includes columns 8 to 10 from df_raw, possibly related to shaft information.
    * cols_speed: This sublist includes columns 11 to 14 from df_raw, likely representing speed-related information.
    * cols_wind: This sublist includes columns 15 to 18 from df_raw, which seem to be related to wind information.
    * cols_sea: This sublist includes columns 19 to 22 from df_raw, which might be related to sea conditions.
    * cols_wave: This sublist includes columns 23 to 25 from df_raw, potentially representing wave-related data.
The df_raw.shape line simply prints the shape of the DataFrame df_raw, indicating the number of rows and columns in the DataFrame.

This code is used to check and visualize missing values in the DataFrame df_raw. Here's an explanation:

* Check missing values:
    * print(df_raw.isnull().sum()): This line calculates and prints the sum of missing values for each column in the DataFrame df_raw. It provides a count of how many missing values are present in each column.
*   msno.matrix:
        * msno.matrix(df_raw): This line creates a matrix visualization of missing values using the msno library. In the matrix:

        * Rows represent different rows (data points) in the DataFrame.
        * Columns represent different columns in the DataFrame.
        * A white line indicates missing data, while a colored line indicates non-missing data. This visualization helps identify patterns of missingness in the dataset, making it easier to decide how to handle missing values during data preprocessing.

* Drop rows with missing target (Main Engine Fuel Consumption) ~ 1% of records:
    * df_mod = df_raw.copy().dropna(): This line creates a new DataFrame df_mod by making a copy of df_raw and then dropping rows with missing values using the dropna() method. This step is performed to remove rows where the target variable ('Main Engine Fuel Consumption') has missing values.
    * df_mod = df_mod.reset_index().drop('index', axis=1): After dropping rows with missing values, this line resets the index of df_mod and drops the old index column.
    * print('Percentage of missing records: ', ((1 - df_mod.shape[0] / n_records) * 100)): This line calculates and prints the percentage of missing records that were removed from the dataset.
    * print('Percentage of remaining records: ', ((df_mod.shape[0] / n_records) * 100)): This line calculates and prints the percentage of remaining records after removing the missing ones.
*   msno.matrix:
    * msno.matrix(df_mod): This line creates a matrix visualization of missing values for the modified DataFrame df_mod. It helps verify that missing values have been effectively removed from the dataset, and the matrix should show fewer missing lines compared to the original matrix.