# Predicting Vessel Navigational Status Using AIS Motion and Ship Specifications: A Case Study in the Kattegat Strait

Rafan Ahmed

ITCS-3156 Final Project Report
Dr. Minwoo (Jake) Lee
December 2025

# Contents

# 1 Introduction

## 1.1 Problem Statement & Motivation

In the contemporary era of maritime shipping activity, the realms of technology and the age-old human ambition of transporting goods over the seas have become deeply intertwined, creating an understated but profoundly real reliance on one another. Real-time information on maritime traffic, provided by Automated Identification System (AIS) data, forms a backbone for modern logistics handling, allowing capital and states to efficiently and effectively allocate resources for both markets and governance. This dependence is especially pronounced among northern European nations, with strategically important bodies of water such as the Kattegat Strait. However, as fundamental as these systems are, AIS messages are often noisy, incomplete, and lacking reliable "navigational status" labels. Maritime analytics, therefore, proves crucial in traffic forecasting, anomaly detection, and port logistics, all of which contribute to maintaining the operational integrity of the many industries and government initiatives that depend on the Kattegat Strait. Technological applications operating primarily on the functional basis of rule-based logic are inferior in a domain defined by the stochasticity of maritime traffic and logistics, particularly within the waters of economically robust nations like those surrounding the Kattegat Strait. Therefore, we aim to develop machine learning applications and models capable of inferring navigational statuses solely from ship movement patterns and specifications.

The motivations for this endeavor are rooted in a deeper but subtle interest in developing quantitative strategies in financial trading. In search of an edge over other agents in the equity and commodity markets, hedge funds use AIS data to estimate commodity flows, shipping demand, congestion at ports, supply chain dynamics and macroeconomic indicators. These signals feed directly into forecasting models for oil prices, freight rates, dry bulk demand, and container throughput. In the context of this report, the Kattegat Strait represents a high-traffic maritime corridor, where major shipping lanes between the North Sea and the Baltic Sea are critical for oil tankers, bulk carriers, and container flows. Disruptions within this region influence freight markets, energy pricing, and even currency movements for Nordic economies. These market-shaping dynamics are crucial for investors and traders trying to properly price assets and allocate capital into efficient channels, and moments of arbitrage arising from informational mismatches create an environment in which possessing an edge on niche data, such as AIS signals from a specific body of water, can produce broader domino effects across European markets.

## 1.2 Project Objectives & Contributions

The objectives of this project are grounded in both the technical challenges of maritime analytics and the broader implications such analytics have for logistics, governance, and quantitative finance. At its core, this work aims to develop supervised machine learning models capable of inferring vessel navigational status solely from AIS-derived motion patterns and ship specifications, providing a data-driven alternative to brittle rule-based systems in a domain defined by stochastic behavior. To achieve this, the project establishes a robust preprocessing pipeline tailored to the realities of AIS data and transforming raw transmissions into a reliable feature space for modeling. This work also makes contributions to analytical insight into which physical and behavioral attributes most strongly determine navigational status and how these distinctions manifest across high-traffic maritime corridors such as the Kattegat Strait. Ultimately, this work delivers a fully reproducible machine learning pipeline, validated models, and a structured repository of code and visualizations that collectively advance both the practical and analytical utility of AIS-based navigational status prediction.

## 2 Data & Domain Background

### 2.1 Automatic Identification System (AIS) Overview

The Automatic Identification System (AIS) is a global maritime communication and tracking framework that was designed to enhance navigational safety, situational awareness, and operational coordination in maritime environments. AIS transponders continuously transmit a stream of dynamic and static information about the vessel from which the AIS tracking originates. The dynamic features of a vessel relayed by an AIS include position, speed on the ground (SOG), course on the ground (COG), while the static features include the type, the length, width and draught of the vessel. Port authorities rely on AIS signals to manage vessel queues, and logistics companies use them to monitor supply chain flows, making AIS a critical data infrastructure for modern maritime analytics. In high-traffic regions such as the Kattegat Strait, where commercial, industrial, and passenger vessels converge, AIS provides real-time visibility into vessel behavior. For machine learning applications, AIS functions as a rich observational record of vessel motion patterns, enabling the inference of behavioral states that are otherwise noisy, inconsistently reported, or operationally ambiguous.

### 2.2 Dataset Description

AIS transmissions collected from vessels transiting the Kattegat Strait between January 1st and March 10th, 2022, were compiled into a CSV file and used for the insights, planning, and modeling carried out in this project. Each observation corresponds to a single AIS message that contains a mixture of static vessel characteristics and dynamic navigational measurements. Although AIS devices can broadcast a broad range of information (IMO number, call sign, destination, and ETA) the dataset only provides the fields most relevant to modeling vessel behavior: ship type, length, width, and draught as static attributes, along with speed over ground (SOG), course over ground (COG), and heading as dynamic indicators. This structured combination of physical specifications and motion-derived features forms the foundation from which we train machine learning models to infer navigational status from observed vessel activity. The composition of the dataset by vessel type is summarized in Figure 1.

### 2.3 Target Variable: Navigational Status

The target variable for this study is the vessel's navigational status, a categorical AIS field that conveys operational state at a given moment. These statuses include signals such as *Under way using engine*, *At anchor*, *Moored*, and *Fishing*, each reflecting a distinct behavioral mode corresponding to a vessel's motion, maneuverability, and operational intent. However, AIS navigational status is notoriously unreliable. It is often entered manually, updates inconsistently, and prone to missing values. This unreliability presents a challenge for stakeholders who rely on accurate maritime awareness, particularly those engaged in equity and commodity pricing, financial trading, and broader market maneuverability. The unreliability also creates a challenge within the domains of machine learning; the noisy and imbalanced distribution of these statuses transforms the task into a complex classification problem. This will require careful preprocessing, stratified splitting, and evaluation metrics that account for class imbalances. Predicting navigational status from observable motion and vessel characteristics serves a dual purpose, compensating for inconsistencies in AIS reporting and enabling behavioral inference directly from movement patterns, independent of human input. The composition of the dataset by navigation status is summarized in Figure 2.

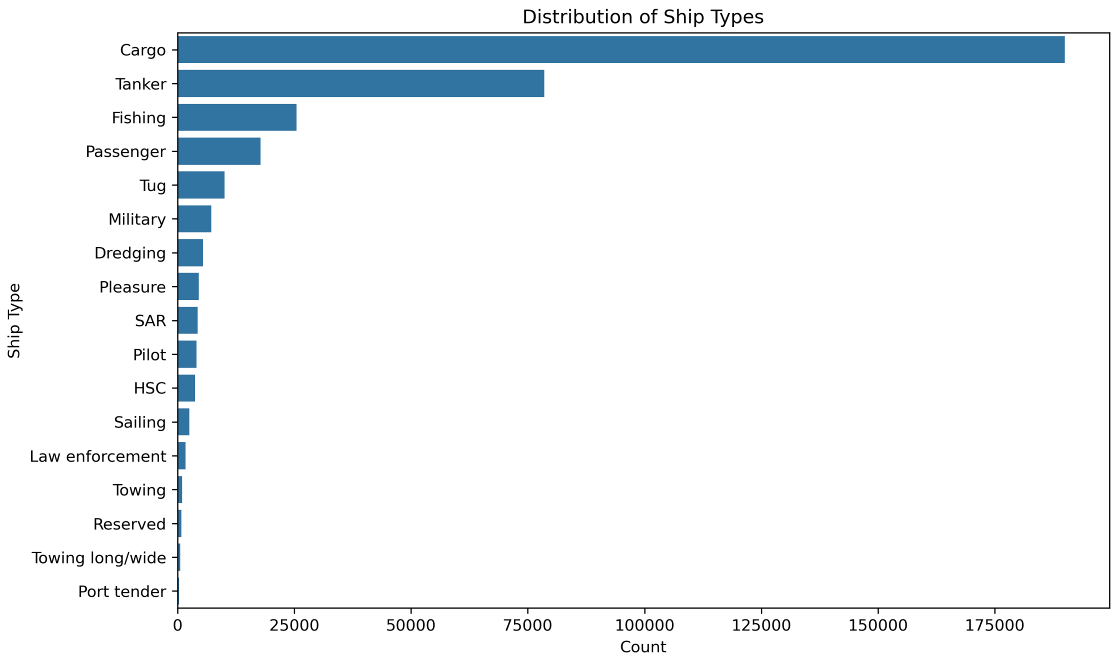


Figure 1: Distribution of vessel types in the AIS dataset for the Kattegat Strait.

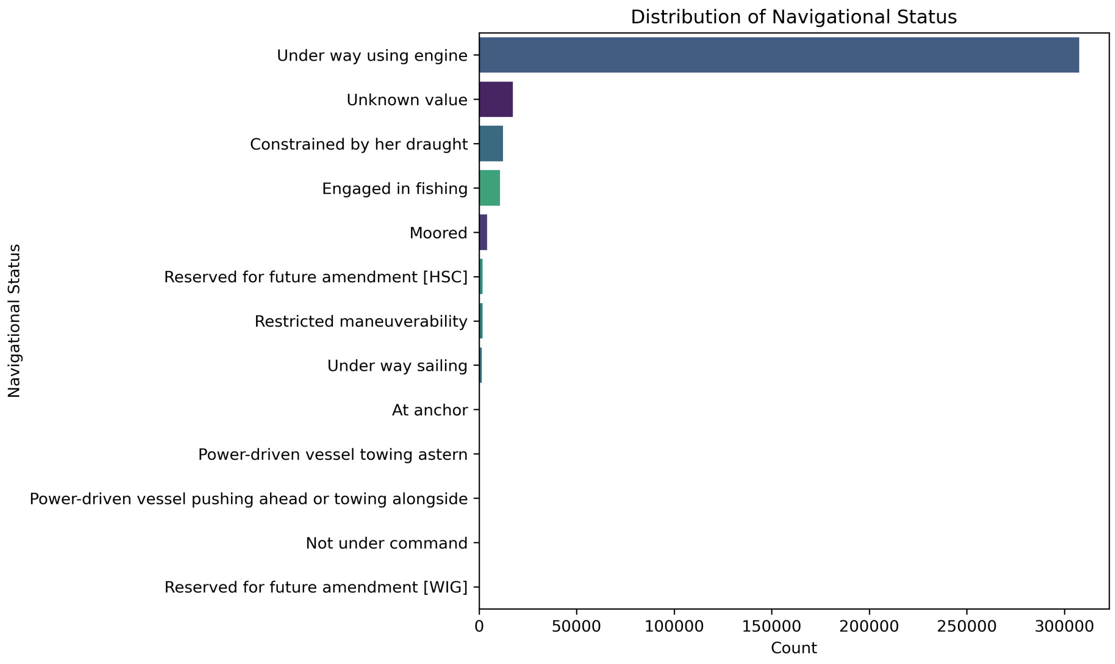
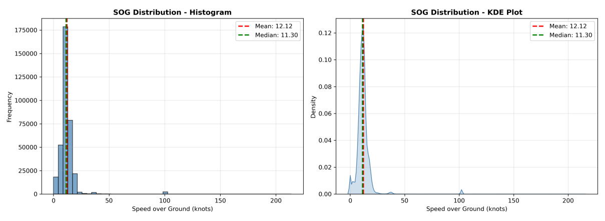


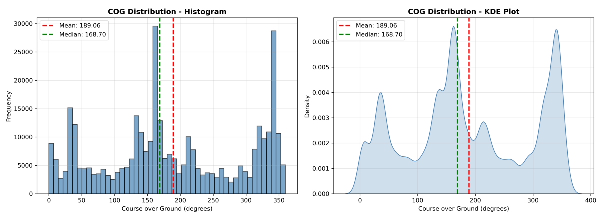
Figure 2: Class distribution of AIS navigational statuses used as the target variable.

## 2.4 Feature Overview: Motion vs Ship Specifications

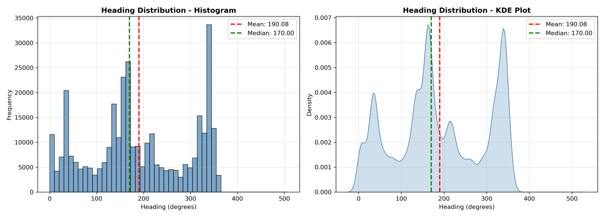
A mix of dynamic motion features and static vessel specifications, each contributing complementary information about vessel behavior, is provided from our dataset. The dynamic features include speed over ground (SOG), course over ground (COG), and heading capture, which are movement patterns that vary significantly across operational states. These motion-derived indicators form the behavioral backbone of the classification task. The static vessel specifications include ship type, length, width, and draught, which encode physical constraints that shape how different classes of vessels operate within a maritime environment. These features provide a representation of vessel identity and motion, enabling machine learning models to infer navigational status from both behavioral and structural cues. The composition of the dataset by empirical distributions of key numeric AIS features and static specifications is summarized in Figure 3.



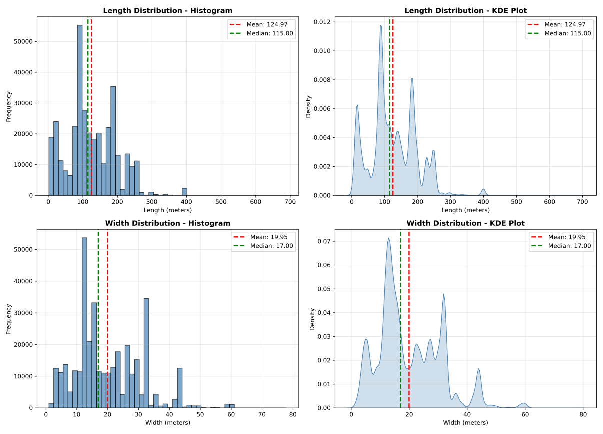
(a) SOG



(b) COG



(c) Heading



(d) Length / Width

Figure 3: Empirical distributions of key numeric AIS features, including motion-related variables (SOG, COG, heading) and static vessel specifications (length, width).

## 2.5 Data Quality, Cleaning, and Leakage Considerations

In this raw dataset, both static vessel attributes and dynamic motion fields contain missing, malformed, or implausible values. These include zero or impossible draught, width, length measurements, contradictory speed-and-heading reports, and jitter-driven direction jumps. Early exploration also indicates strong class imbalance: *Under way using engine* dominates, while states like *Fishing* or *Restricted maneuverability* are rare, risking skewed models. Visualizations further reveal tight links between vessel IDs (e.g., MMSI) and physical attributes, creating leakage risks where a model learns vessel identity rather than behavior. In essence, missingness, implausible values, noise, class imbalance, and identity leakage all plague this dataset. The preprocessing pipeline should aim to retain true behavioral signals while removing artifacts that would undermine model validity.

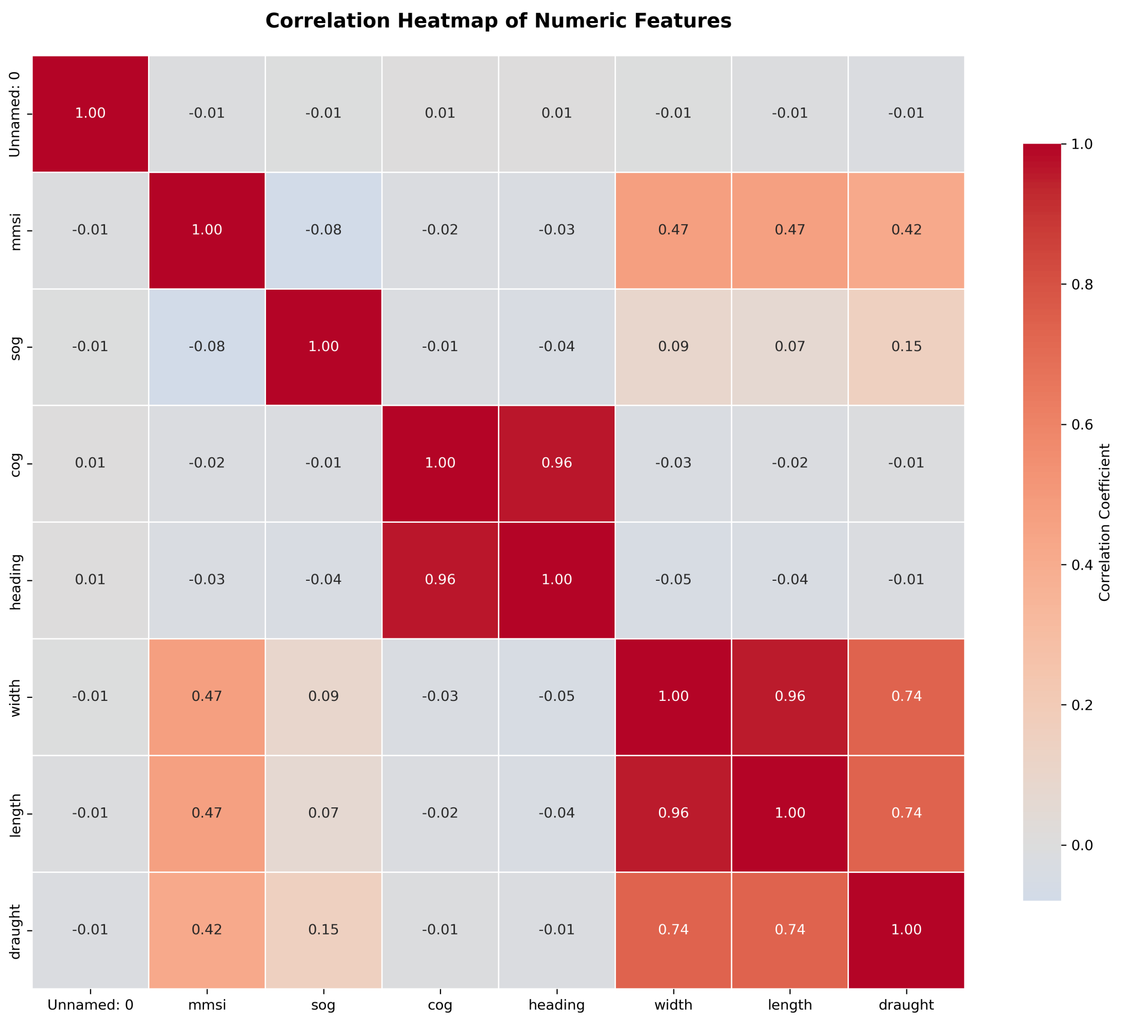
## 3 Methodology

### 3.1 Preprocessing Pipeline

A first step involved removing identifier-based leakage sources. The MMSI field, along with other static identifiers tied to vessel identity, exhibited strong correlations with physical ship dimensions such as length, width, and draught. Including these fields would allow a model to implicitly memorize vessel identity rather than infer navigational behavior, undermining generalization. As a result, all identity-bearing fields were excluded from the feature set. Next, the dataset was subjected to domain-informed physical filtering to eliminate implausible observations. AIS messages occasionally reported nonsensical speeds, invalid headings (e.g., negative or $> 360^\circ$ values), or impossible draught measurements. Records containing these violations were removed to maintain physical coherence within the feature space. Additional type standardization resolved inconsistencies in categorical fields (such as ship type) caused by formatting irregularities in the raw transmissions.

The pipeline then addressed missing, duplicated, and malformed entries. Rows with null values in any of the core modeling features, such as SOG, COG, heading, draught, length, or width, were removed, as imputing them would introduce distortions into variables tied closely to vessel physics. Duplicate AIS messages were also dropped to prevent overweighting particular vessel trajectories. Following cleaning, continuous features were normalized into comparable numeric scales to stabilize model training, particularly for algorithms sensitive to feature magnitude. Categorical labels, such as ship type, were encoded into numerical representations suitable for downstream classifiers. These transformations ensured that both motion-derived and physical characteristics contributed proportionally to the learning process.

All preprocessing operations were implemented within a consistent and reproducible framework across the training, validation, and test splits, applied later in the methodology. This structured approach produces a refined dataset in which the remaining samples reflect realistic maritime behavior and are free from identity leakage. The unprocessed feature correlation matrix with null values and MMSI data leakage is shown on Figure 4.



|  | Unnamed: 0 | mmsi | sog | cog | heading | width | length | draught |
|---|---|---|---|---|---|---|---|---|
| Unnamed: 0 | 1.00 | -0.01 | -0.01 | 0.01 | 0.01 | -0.01 | -0.01 | -0.01 |
| mmsi | -0.01 | 1.00 | -0.08 | -0.02 | -0.03 | 0.47 | 0.47 | 0.42 |
| sog | -0.01 | -0.08 | 1.00 | -0.01 | -0.04 | 0.09 | 0.07 | 0.15 |
| cog | 0.01 | -0.02 | -0.01 | 1.00 | 0.96 | -0.03 | -0.02 | -0.01 |
| heading | 0.01 | -0.03 | -0.04 | 0.96 | 1.00 | -0.05 | -0.04 | -0.01 |
| width | -0.01 | 0.47 | 0.09 | -0.03 | -0.05 | 1.00 | 0.96 | 0.74 |
| length | -0.01 | 0.47 | 0.07 | -0.02 | -0.04 | 0.96 | 1.00 | 0.74 |
| draught | -0.01 | 0.42 | 0.15 | -0.01 | -0.01 | 0.74 | 0.74 | 1.00 |

Figure 4: *MMSI* exhibits moderate correlations (r = 0.42–0.47) with vessel dimensions, indicating data leakage that requires its removal before modeling. *Unnamed: 0* are null entries in the dataset that must be removed.

## 3.2 Train/Validation/Test Split Strategy

In the pursuit of building reliable machine learning models for navigational status classification, it is essential to ensure that the data splits used during training, hyperparameter tuning, and final evaluation accurately reflect the true class distribution of the dataset. In our dataset, AIS navigational statuses are highly imbalanced. *Under way using* represents over 91% of all transmissions, and several operational classes account for less than 1%. A naive random split would risk producing validation and test sets that lack representation from minority categories. This would artificially inflate performance and hinder the model's ability to generalize to rare but operationally important vessel behaviors. To address this, the project employs a two-stage stratified splitting strategy, implemented through a custom `train_val_test_split()` function in a Python preprocessing file (`preprocessing.py`):

1. **Stage 1 – Train/Validation vs. Test (80/20 split):**
   The first split allocates 20% of the data as a held-out test set, untouched until final evaluation.

Stratification ensures that the navigational status distribution is preserved, preventing the test set from being dominated by majority classes.

2. **Stage 2 – Train vs. Validation (80/20 of the remaining 80%):**
   The remaining 80% (Train + Validation) is then split again such that:
   - 64% of the total data becomes the training set,
   - 16% becomes the validation set, used exclusively for hyperparameter tuning and model selection.

This two-step approach yields the final 64/16/20 distribution, as documented in the modeling notebook, with class proportions in each split matching the global distribution nearly identically (e.g., *Under way using engine* $\approx 0.919$ in train, validation, and test).

## 3.3 Classification Models

The subsections below present the mathematical foundations, interpretive components, and motivation for these models in this navigational-status classification task.

### 3.3.1 Logistic Regression

Logistic Regression provides a linear baseline for multiclass classification via the softmax formulation. For an input feature vector $\mathbf{x} \in \mathbb{R}^d$ and $K$ navigational-status classes, the model estimates class probabilities:

$$P(y = k \mid \mathbf{x}) = \frac{\exp(\boldsymbol{\theta}_k^\top \mathbf{x})}{\sum_{j=1}^{K} \exp(\boldsymbol{\theta}_j^\top \mathbf{x})}.$$

The predicted class is:

$$\hat{y} = \arg\max_k \; P(y = k \mid \mathbf{x}).$$

Where,

- $\mathbf{x} \in \mathbb{R}^d$ – the feature vector for a single AIS message, containing all cleaned vessel features; each row of the model-ready CSV corresponds to one such $\mathbf{x}$.
- $\boldsymbol{\theta}_k \in \mathbb{R}^d$ – the learned weight vector for class $k$ (e.g., *Under way using engine*, *At anchor*), where each entry indicates how a feature in $\mathbf{x}$ shifts the log-odds of class $k$.
- $\boldsymbol{\theta}_k^\top \mathbf{x}$ – the linear score for class $k$, computed as a weighted sum of AIS features, with large positive weights (e.g., on SOG) increasing the score for motion-heavy classes.
- $K$ – the number of navigational-status classes after removing ambiguous or extremely rare labels.
- $P(y = k \mid \mathbf{x})$ – the predicted probability that a vessel with features $\mathbf{x}$ is in class $k$, used for loss computation and evaluation metrics such as macro-F1.
- $\exp(\cdot)$ – the exponential function applied to class scores to make them positive and enhance separability before softmax.

- $\sum_{j=1}^{K} \exp(\boldsymbol{\theta}_j^\top \mathbf{x})$ – the normalization term ensuring the softmax probabilities sum to 1 across all classes.
- **Softmax function** – the transformation from linear scores $\boldsymbol{\theta}_k^\top \mathbf{x}$ to probabilities $P(y = k \mid \mathbf{x})$, converting motion patterns into interpretable class likelihoods.
- $\hat{y} = \arg\max_k P(y = k \mid \mathbf{x})$ – the final predicted status, selecting the class with highest probability (e.g., high SOG and stable COG often predict *Under way using engine* over stationary classes).

Logistic regression assumes decision boundaries that are linear in the feature space. This makes the model fast, interpretable, and valuable as a baseline; however, it may struggle with non-linear vessel-motion patterns.

### 3.3.2 Random Forest

Random Forest is an ensemble of $B$ decision trees trained on samples of the data, where each tree learns a different approximation of the relationship between features and the target, and the final class prediction is obtained by majority vote across all trees:

$$\hat{y} = \text{mode}\big(h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_B(\mathbf{x})\big).$$

Each decision tree $h_b$ recursively splits the feature space by selecting the feature $f$ and threshold $t$ that yield the largest impurity reduction from the parent node to its children:

$$\Delta I = I(\text{parent}) - \left(\frac{N_L}{N} I(\text{left}) + \frac{N_R}{N} I(\text{right})\right),$$

with common impurity measures defined as:

$$I_{\text{Gini}}(p) = 1 - \sum_{k=1}^{K} p_k^2, \qquad I_{\text{Entropy}}(p) = -\sum_{k=1}^{K} p_k \log p_k.$$

Where,

- $h_b$ – the $b$-th decision tree, trained on its own bootstrap sample so that each tree learns slightly different patterns because no two trees receive the exact same training data.
- $B$ – total number of trees in the forest, where increasing $B$ typically improves stability and predictive accuracy by reducing variance through the aggregation of many independent trees.
- $\mathbf{x}$ – the input feature vector for a vessel AIS message, containing SOG, COG, heading, draught, width, and one-hot ship type encodings, all of which are used by each tree to make split decisions.
- $I(\cdot)$ – the impurity function that quantifies how mixed the classes are within a node, with lower impurity indicating that the samples in the node belong mainly to one class.
- $N_L, N_R$ – the number of samples falling into the left and right child nodes after a split, where $N = N_L + N_R$ represents the total number of samples in the parent node.
- Randomization – Random Forest injects randomness in two ways:
  - **Bootstrap sampling**: each tree receives a different sampled subset of the rows.

  - **Random feature selection**: each split only considers a random subset of available features.

These mechanisms reduce correlation between trees and prevent overfitting.

Random Forests build a strong and stable model by combining many simple decision trees into an ensemble capable of learning complex nonlinear patterns, handling mixed feature types, tolerating outliers and noisy AIS measurements, and capturing interactions such as the relationship between draught and vessel speed, making the method well-suited for AIS data, where noise, irregular readings, and variable vessel behavior are common.

### 3.3.3 XGBoost (Extreme Gradient Boosting)

XGBoost is a gradient-boosted decision tree model that builds trees sequentially, with each new tree correcting the remaining errors of the model so far, forming an additive predictor rather than training independent trees like Random Forest:

$$\hat{y}^{(t)} = \sum_{i=1}^{t} f_i(\mathbf{x}),$$

Each tree $f_t$ is trained to approximate the negative gradient and curvature (second-order information) of the loss with respect to current predictions, so the tree added at stage $t$ is chosen by:

$$f_t = \arg\min_{f} \sum_{i=1}^{N} \ell\big(y_i,\ \hat{y}_i^{(t-1)} + f(\mathbf{x}_i)\big) + \Omega(f),$$

and the regularization term:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

penalizes models that grow trees with too many leaves or overly large leaf weights.
Where,

- $\hat{y}^{(t)}$ – the model's cumulative prediction after $t$ trees have been added, representing a vector of log-odds across classes in a multiclass navigational-status setting.
- $f_t$ – the decision tree learned at iteration $t$, which is trained specifically to reduce the errors that the previous ensemble has not yet resolved.
- $\ell$ – the multiclass logistic loss function used in this project to quantify how accurately the model predicts the vessel's navigational status.
- $\Omega(f)$ – the regularization term that limits tree complexity by penalizing both the number of leaves ($T$) and the magnitude of leaf weights ($w_j$), making XGBoost more resistant to overfitting.
- $T$ – the number of terminal nodes in a tree, where small $T$ creates simpler models and larger $T$ enables capturing richer nonlinear patterns.

- $w_j$ – the logit score assigned to leaf $j$, which shifts the model's output whenever a sample lands in that leaf.
- $\gamma$ – the cost of adding a new leaf, with larger values discouraging deep or overly complex trees.
- $\lambda$ – the L2 regularization parameter that shrinks leaf weights, improving stability and reducing noise sensitivity in AIS features.

XGBoost differs from Random Forest in its utilization of second-order optimization (gradients) for more precise updates. This includes built-in regularization through $\gamma$ and $\lambda$ to prevent overly complex trees, and naturally focuses on samples difficult to classify, making it effective for imbalanced classes. These features allow XGBoost to capture subtle patterns in AIS signals that may be overlooked by simpler models.

## 3.4 Evaluation Metrics

The classification problem in this project is highly imbalanced, with the majority class (*Under way using engine*) comprising approximately 92% of all AIS messages and several minority statuses representing less than 1% of the dataset (see distributions in the modeling notebook). Therefore, relying on accuracy alone would paint a misleading picture of model performance. To obtain a more comprehensive, class-sensitive evaluation, we compute the following metrics for all models:

- Accuracy
- Macro-Averaged Precision
- Macro-Averaged Recall
- Macro-Averaged F1 Score
- Weighted F1 Score
- Confusion Matrix (normalized)

### 3.4.1 Accuracy

Accuracy measures the proportion of correct predictions:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total predictions}}.$$

Accuracy provides an overall performance summary but is not reliable under class imbalance, since a model that predicts the majority class nearly always would still achieve more than 90% accuracy in this dataset.

### 3.4.2 Precision (Macro)

Macro-averaged precision treats all classes equally by computing precision per class and then averaging. This metric captures the model's ability to avoid false positives across all navigational statuses.

$$\text{Precision}_{\text{macro}} = \frac{1}{K}\sum_{k=1}^{K}\frac{TP_k}{TP_k + FP_k}.$$

#### 3.4.3 Recall (Macro)

Macro recall averages the per-class recall. This metric captures the model's ability to avoid false positives across all navigational statuses.

$$\text{Recall}_{\text{macro}} = \frac{1}{K}\sum_{k=1}^{K}\frac{TP_k}{TP_k + FN_k}.$$

#### 3.4.4 F1 Score

We report both macro-averaged and weighted F1. Macro-F1 punishes poor performance in any single class and is the primary metric of interest for this imbalanced maritime dataset.

$$\text{F1}_{\text{macro}} = \frac{1}{K}\sum_{k=1}^{K}\frac{2 \cdot \text{Precision}_k \cdot \text{Recall}_k}{\text{Precision}_k + \text{Recall}_k}.$$

Weighted F1 accounts for class frequency and reflects performance weighted toward the majority class.

$$\text{F1}_{\text{weighted}} = \sum_{k=1}^{K} w_k \cdot \text{F1}_k, \quad w_k = \frac{N_k}{N}.$$

#### 3.4.5 Confusion Matrix (Normalized)

Confusion matrices, normalized row-wise, are generated for each model. These matrices show which classes the model confuses, how each model behaves differently, and how well motion-based features like SOG, COG, and heading distinguish navigational statuses.

$$\text{CM}_{ij} = \frac{\text{Number of samples from true class } i \text{ predicted as } j}{\text{Total samples in true class } i}.$$

## 4 Experiments & Results

### 4.1 Experimental Setup

The experimental workflow for this project follows a standardized machine learning pipeline designed to ensure reproducibility, fairness across models, and robustness against the heavy class imbalance present in AIS navigational data. All experiments were conducted using the preprocessed dataset generated earlier in the project, where feature engineering, cleaning, and anti-leakage measures were applied. The following components characterize the experimental setup:

#### 4.1.1 Dataset Splitting and Inputs

All models were trained on the same **stratified** train/validation/test splits produced by the custom `train_val_test_split()` function. The final distribution is:

- 64% training data
- 16% validation data
- 20% held-out test data

Stratification ensured that rare navigational statuses (e.g., *Moored*, *At anchor*, *Towing*) were represented proportionally across all three sets, preventing model bias toward the overwhelming majority class (*Under way using engine*). The input features used for all models include:

- Dynamic motion features: SOG, COG, heading
- Static vessel features: length, width, draught
- One-hot–encoded ship-type categories

Identifier fields (e.g., MMSI) were excluded to prevent leakage.

### 4.1.2 Model Construction

All models were instantiated using modular builder functions defined in `train_models.py`:

- Logistic Regression – baseline linear classifier
- Random Forest – nonlinear ensemble baseline
- XGBoost – gradient-boosted decision tree model

Each model was trained only on the training split and evaluated first on the validation split for comparison. Hyperparameters remained at reasonable defaults to maintain comparability and avoid overfitting through excessive tuning.

### 4.1.3 Training Procedure

1. Fit the model on the training set.
2. Generate predictions for the validation set.
3. Compute evaluation metrics using the shared `evaluate_model()` helper:
    - accuracy
    - macro precision
    - macro recall
    - macro F1
    - weighted F1
4. Record confusion matrices and classification reports.
5. Repeat process from the beginning across all models.

### 4.1.4 Tracking Validation Performance

Each model's validation performance was logged into a CSV file (`validation_metrics.csv`), enabling transparent comparison. Confusion matrices and per-class precision-recall patterns were also saved, allowing deeper analysis of class-specific performance.

### 4.1.5 Final Evaluation on the Held-Out Test Set

After identifying Random Forest and XGBoost as the top-performing models from validation results, the models were evaluated on the completely unseen 20% test set. Metrics and confusion matrices were exported to:

- `model_evaluation_metrics.csv`
- `classification_report_{model}_test.txt`
- `confusion_matrix_{model}_test.csv`

This ensures full reproducibility and transparency in reporting.

### 4.1.6 Computational Environment

All experiments were executed in Python using:

- `scikit-learn` for baseline models
- `xgboost` for gradient boosting
- `pandas/numpy` for data operations
- `matplotlib/seaborn` for visual diagnostics

The environment was consistent across all training runs to ensure comparability.

## 4.2 Validation Results

Model performance was first assessed on the validation split (16% of the full dataset), using accuracy, macro-F1, weighted-F1, macro-precision, and macro-recall as defined in Section 3.4. All metrics were computed using the standardized evaluation function described in the modeling notebook and recorded in `validation_metrics.csv`. Random Forest demonstrated the best validation performance, indicating effective generalization and strong handling of AIS motion features. XGBoost also performed well, while Logistic Regression showed clear limitations on minority classes.

Table 1: Validation Set Performance Metrics for All Models

| **Model** | **Accuracy** | **Macro-F1** | **Weighted-F1** | **Precision (Macro)** | **Recall (Macro)** |
|---|---|---|---|---|---|
| Logistic Regression | 0.717 | 0.373 | 0.797 | 0.303 | 0.784 |
| Random Forest | 0.986 | 0.875 | 0.985 | 0.915 | 0.844 |
| XGBoost | 0.978 | 0.834 | 0.978 | 0.886 | 0.800 |

## 4.3 Test Results

On the held-out 20% test set, all three models generalize consistently with their validation performance. Random Forest remains the strongest model overall, achieving the highest accuracy and macro-F1, while XGBoost trails only slightly. Logistic Regression performs noticeably worse on macro-averaged metrics, indicating that a linear decision surface struggles to capture the minority navigational statuses compared to the tree-based methods.

Table 2: Test set performance metrics for all models

| Model | Accuracy | Macro-F1 | Weighted-F1 | Precision (Macro) | Recall (Macro) |
|---|---|---|---|---|---|
| Logistic Regression | 0.719 | 0.369 | 0.798 | 0.300 | 0.772 |
| Random Forest | 0.985 | 0.883 | 0.985 | 0.919 | 0.860 |
| XGBoost | 0.979 | 0.846 | 0.978 | 0.884 | 0.826 |

## 4.4 Confusion Matrix Analysis

Logistic Regression struggles to separate several navigational statuses, especially those with nonlinear or overlapping motion patterns. The confusion matrix shows frequent errors between *At anchor*, *Moored*, and *Restricted maneuverability*, as well as mixing between *Under way sailing* and *Under way using engine*. These issues arise because linear decision boundaries cannot capture the nonlinear structure of AIS motion data, where subtle variations in speed, heading stability, and maneuvering behavior are essential. Thus, Logistic Regression functions mainly as a baseline rather than a competitive model.

XGBoost provides much stronger discrimination, particularly for minority classes such as *Engaged in fishing*, *Constrained by her draught*, and *Restricted maneuverability*, achieving near-perfect recall in several cases. Some confusion remains between *At anchor* and *Moored* due to their inherently similar low-speed motion profiles. XGBoost models nonlinear relationships between ship type, draught, and motion features more effectively than Logistic Regression. However, it still slightly trails Random Forest on high-volume classes, such as *Under way using engine*.

Random Forest delivers the most accurate and stable performance overall, with near-perfect classification for classes such as *Engaged in fishing*, *Power-driven vessel towing astern*, and *HSC*. It performs well even on low-sample classes, showing robustness to imbalance and noise. Remaining confusion between *At anchor* vs. *Moored* and *Under way sailing* vs. *Under way using engine* reflects genuine similarities in vessel motion. Random Forest captures the nonlinear structure of navigational behavior more effectively than the other models. Figure 5 shows the confusion matrix for Random Forest.

## 4.5 Feature Importance Analysis

Feature importance was evaluated for the two tree based models: Random Forest and XGBoost. Both methods provide insight into which AIS attributes contribute most to predicting a vessel's navigational status.

Random Forest places the highest importance on physical motion and vessel size features. Draught, length, speed over ground, and width appear at the top of the ranking. These variables describe how large a vessel is and how it is moving, which directly shapes whether a vessel is under way, moored, or at anchor. Several ship type categories also carry moderate importance and help the model distinguish specialized movements such as fishing or tug operations.

XGBoost shows a different pattern. Ship type indicators dominate the top positions, especially Fishing, HSC, and Towing long or wide. XGBoost relies more on categorical identity to infer the likely motion pattern of a vessel. Physical variables such as draught and speed still matter, but they appear below the most informative ship type categories.

Together, these results show that vessel motion features and ship type categories both carry meaningful predictive signal. Random Forest depends more on physical movement, while XGBoost depends more on the operational purpose of the vessel. Figure 6 shows the feature importance for

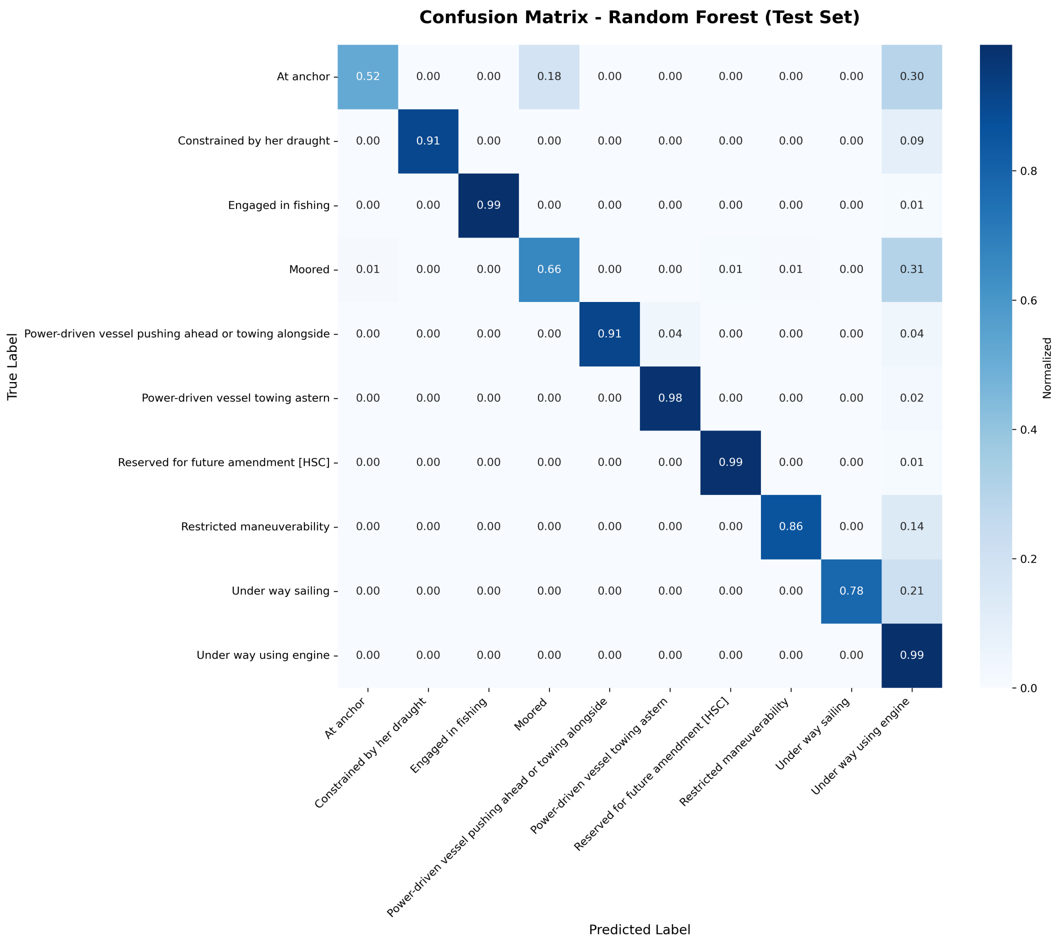


Figure 5: Confusion Matrix for Random Forest Classification.

Random Forest.

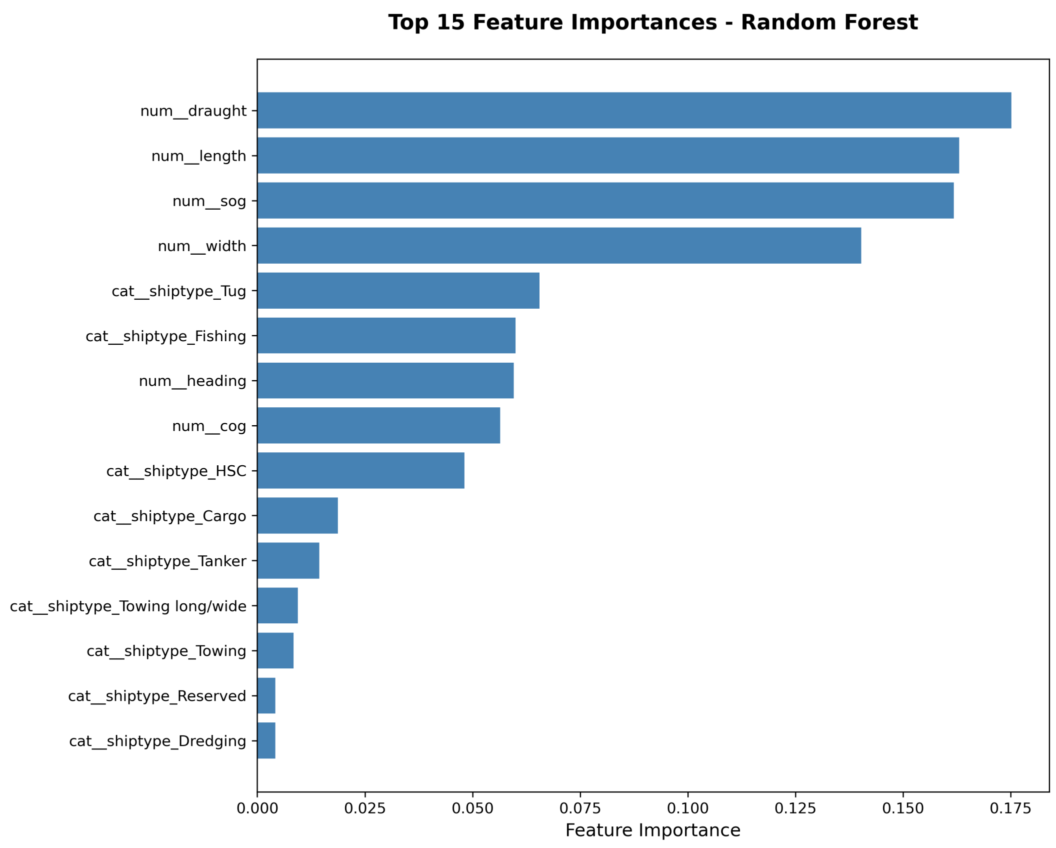


Figure 6: Feature Importance for Random Forest Classification.

## 5 Discussion

### 5.1 Impact of AIS Features on Status Prediction

AIS feature importance patterns show that vessel behavior in the Kattegat Strait carries a clear, learnable structure. Motion variables, such as SOG, COG, heading, draught, and vessel size, were the strongest predictors because they directly reflect how ships operate within this narrow, high-traffic corridor. The models captured realistic maritime behavior (e.g., near-zero speeds for anchored vessels, distinct movement for fishing vessels, slower maneuvering for large ships), while ship-type categories added meaningful operational context, especially for XGBoost. For quantitative applications, shifts in navigational status can indicate port congestion, bottlenecks, and disruptions in commodity flow, which are inputs often used in forecasting energy markets and European supply-chain conditions. Overall, the results confirm that AIS data is multi-modal, economically informative, and well-suited for ML-driven interpretation of maritime activity.

### 5.2 Limitations

Several limitations affect the results. AIS data is noisy and inconsistent, as reports may be missing, inaccurate, or delayed. This was something that was explicitly mentioned and dealt with in the preprocessing step. Navigational status labels themselves are not always reliable because they are manually entered by the crew, additionally introducing human error into training and evaluation. The models also treat each AIS message independently, without modeling sequence or time. Real vessel behavior evolves over time, so temporal approaches could yield better accuracy. Finally, while tree-based models perform well, their interpretability is limited, and the dataset represents only one region and time frame. These constraints mean the results should be viewed as strong within-sample performance, not necessarily as generalized operational readiness.

## 6 Conclusion

This project shows that AIS data can be used to accurately predict vessel navigational status using modern machine-learning methods. With combined and careful preprocessing, leakage prevention, stratified splitting, and evaluation across multiple models, the study demonstrates that non-linear models, such as Random Forest and XGBoost, capture complex maritime behavior patterns far better than linear and rule-based baselines. More broadly, the results highlight how physical vessel dynamics and structural ship characteristics interact to form predictive signals, which illustrates how domain knowledge, quantitative analysis, and ML can work together. Although limitations remain, the framework developed here provides a solid foundation for future work involving temporal modeling, anomaly detection, or real-time maritime monitoring.

## 7 References

## References

[1] Erdonmez, Emin Serkan. *AIS Dataset*. Kaggle, 2022, https://www.kaggle.com/datasets/eminserkanerdonmez/ais-dataset?select=ais_data.csv.

[2] Arslanalp, Serkan, et al. "Big Data on Vessel Traffic: Nowcasting Trade Flows in Real Time." *IMF Working Paper*, WP/19/275, International Monetary Fund, 2019, pp. 1–34. *imf.org*.

[3] SeaVantage. "How AIS Streamlines Global Trade: Real-Time Insights & Efficiency." *SeaVantage Blog*, 10 Dec. 2024. *seavantage.com*.

[4] Mapes, Terri. "The Kattegat: What Is It?" *ThoughtCo*, 23 Apr. 2025. *thoughtco.com*.

[5] Mukhopadhyay, Inder. "AIS Navigation Status Codes." *Scribd*, n.d. *scribd.com*.

[6] Yang, Yinchen, et al. "Harnessing the Power of Machine Learning for AIS Data-Driven Maritime Research: A Comprehensive Review." *Transportation Research Part E*, vol. 183, 2024, article 103426. *research.chalmers.se*.

[7] Team WAKE (Worldwide AIS Network). "AIS Data for Commodity Trading: Turning Ship Movements into Market Signals." 2 Oct. 2035. *worldwideais.org*.

[8] Verschuur, Jasper, et al. "Systemic Impacts of Disruptions at Maritime Chokepoints." *Nature Communications*, vol. 16, 2025, article 10421. *nature.com*.

[9] Lee, Jake, and Hongfei Xue. "M09_Logistic Regression." *Intro to Machine Learning*, Department of Computer Science, University of North Carolina at Charlotte, 2025.

[10] Chen, Tianqi, and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System." *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794.

[11] Breiman, Leo. "Random Forests." *Machine Learning*, vol. 45, no. 1, 2001, pp. 5–32.

[12] The Pandas Development Team. *Pandas Documentation.* pandas.pydata.org, 2025, https://pandas.pydata.org/docs/.

[13] Pedregosa, Fabian, et al. "Scikit-learn: Machine Learning in Python." *Journal of Machine Learning Research*, vol. 12, 2011, pp. 2825–2830.

[14] Scikit-learn Developers. *Scikit-learn Documentation.* scikit-learn.org, 2025, https://scikit-learn.org/stable/documentation.html.

[15] Chen, Tianqi, and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System." *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794.

## Acknowledgements

The mathematical explanations, model interpretations, and analytical reasoning presented in this report were informed by a combination of materials from my Intro to Machine Learning course, including lecture notes, course modules, and summaries I organized using NotebookLM, which helped consolidate my own understanding of the material. This project additionally drew on several external scholarly and technical resources, listed in the References section, which were used to support the mathematical rigor and contextual grounding of the models studied. These resources informed my understanding but did not generate any content on my behalf. I also made use of Grammarly to improve writing clarity and Overleaf for LaTeX formatting throughout the document. Large-model assistance was used only for LaTeX structuring, such as formatting tables, equations, and figures. All conceptual explanations, maritime interpretations, feature analyses, and modeling discussions were written entirely on my own.

## Source Code

https://github.com/rafanahmed/Maritime-Navigational-Status-Classification