Emil Skogheim, Henrik Månum, Lars Talian Stangebye-Hansen

Prediction of future vessel coordinates using Machine learning methods on AIS data.

2024-11-10

NTNU

Norwegian University of Science and Technology

Table of content

1. Introduction

With the advent of big data and machine learning, predictive modeling has become instrumental in advancing maritime safety, efficiency, and navigation management. Automatic Identification System (AIS) data, which captures dynamic information on vessel positions, courses, speeds, and other movement characteristics, serves as an essential tool in anticipating vessel trajectories and preventing potential hazards at sea.

This study aims to develop and evaluate machine learning models for predicting vessel coordinates from May 8th to May 12th, using AIS data collected between January 1st and May 7th. As part of the TDT4173 machine learning course, this project emphasizes both theoretical and practical aspects of machine learning, specifically applied to time-series forecasting. The predictive task utilizes historical AIS data to capture temporal and spatial patterns, thereby enabling us to forecast vessel locations.

To support model development and validation, the following datasets are employed:

- ais_train.csv: Historical AIS data with time-series information on vessel positions and movement.

- ais_test.csv: A dataset representing future vessel positions to be predicted, acting as the primary test set.
- vessels.csv: Supplementary data on vessel characteristics, such as size and type, which provide additional predictive context.
- ports.csv: Geographic coordinates of ports, supplying spatial reference points.
- schedules_to_may_2024.csv: Vessel schedules that offer temporal benchmarks and insights into movement patterns.

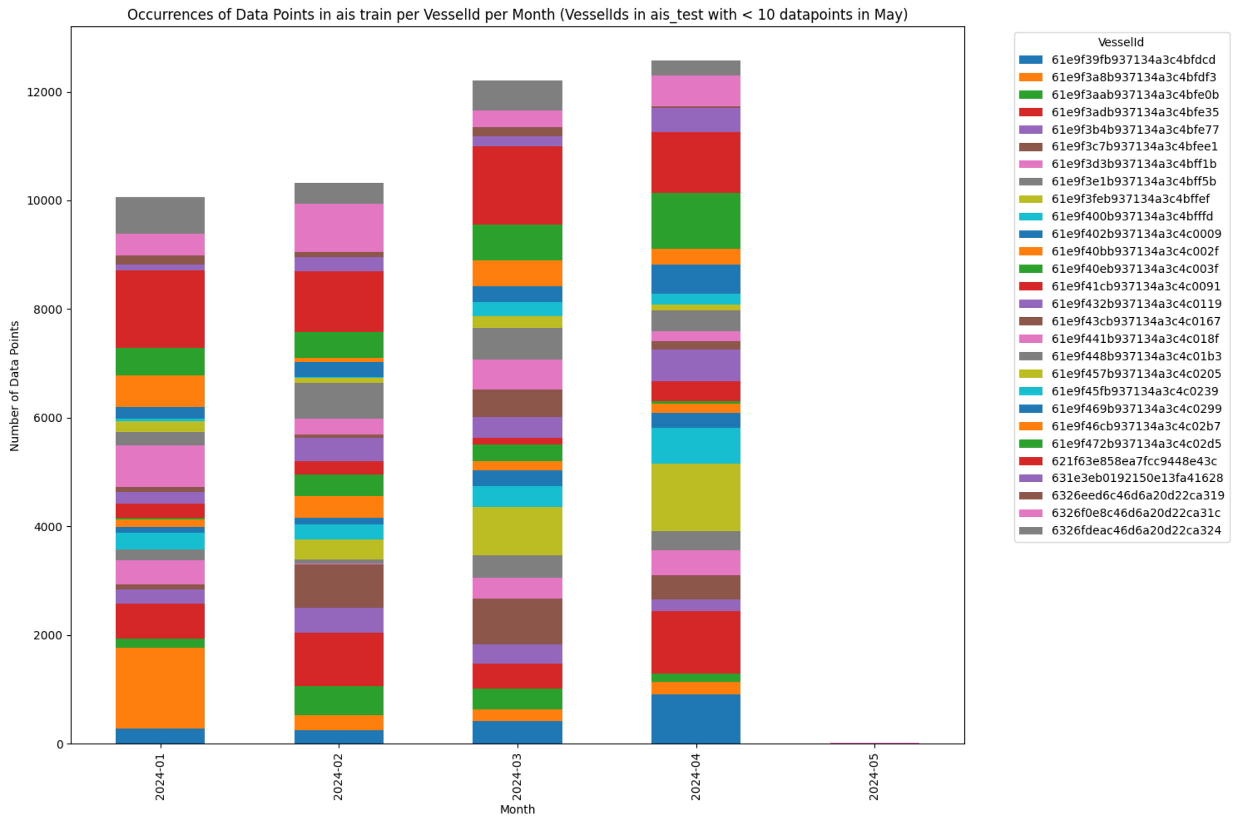
Our approach begins with a comprehensive exploratory data analysis (EDA) to identify the primary characteristics, anomalies, and patterns within the AIS dataset. Next, we employ robust data preprocessing and feature engineering strategies to address inconsistencies, such as missing values and irregular time intervals, and to construct meaningful features that capture vessel behavior. The feature set includes lagged variables, interaction terms, and proximity indicators, designed to improve predictive accuracy.

The model development phase explores various machine learning techniques, including regression models and deep learning approaches, to evaluate their effectiveness in predicting vessel positions. Through a comparison of models based on Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), we aim to balance model interpretability, computational efficiency, and prediction accuracy within the constraints of the project.

By employing machine learning to predict vessel trajectories, this study attempts to contribute to the broader field of maritime data science. In the future, integrating additional contextual data—such as real-time weather and sea conditions—could further improve the accuracy and applicability of vessel position forecasts. This report details our methodology, experimental results, and recommendations, forming a comprehensive approach to the predictive modeling of AIS data in maritime applications.

2. Exploratory Data Analysis (EDA)

The ais train dataset provided contains time series vessel data from 1st of January to 7th of May 2024. Along with the vessels longitude and latitude the dataset contains key features of the vessel movement like speed over ground (SOG), rate of turn (ROG), course over ground (COG) and its heading. These features can be crucial for accurately predicting the future positions of a vessel. The ais test dataset contains a subset of the vessels in the train dataset, and without any features or knowledge of the ship at that given time. The test dataset begins from 00:00 on the 8th of May, meaning that our model would need to predict t+5 days (where t is the last to time known to the model) into the future with hourly timestamps. an excellent case for a deep learning approach with LSTM or GRU. However, after thorough analysis of the dataset, the prediction problem was not that simple.

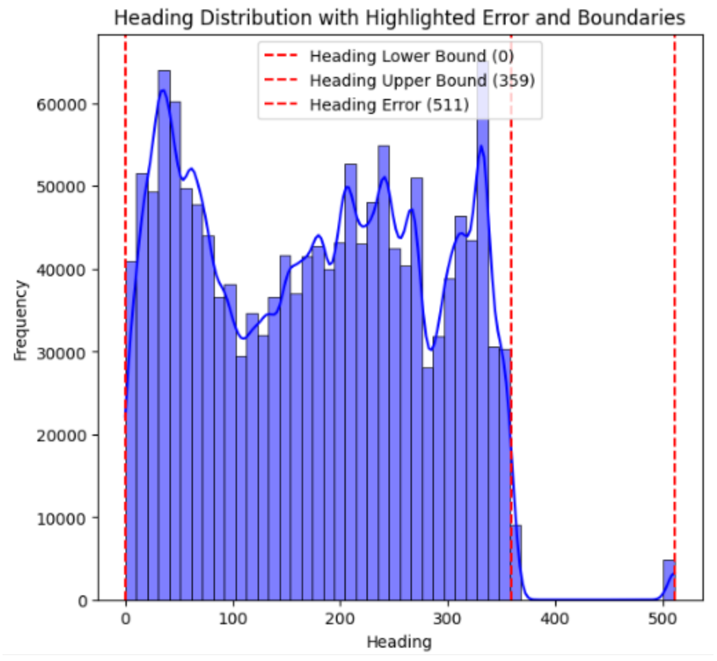


Vessels with less than 10 datapoints in May that is present in ais test visualized as a stacked bar graph. (Appendix 1)

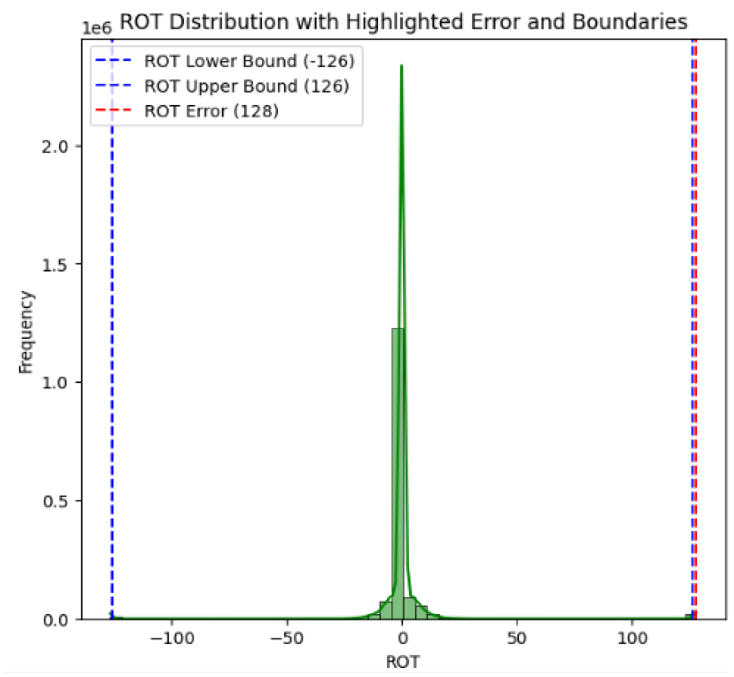
As seen in the graph above, the ais train data set was far from perfect. 20 out of 215 vessels in the ais test dataset had no data in May at all. The challenge of predicting the positions of the vessels from 8th to the 12th of may is no longer just predicting the vessel positions 5 days into the future, as the data availability of the vessels were highly diverse. Furthermore, the datapoints in the train and test set were unevenly spaced, with some vessels containing up to 3-4 datapoints per hour, while some did not have datapoints for weeks at a time. Firstly, the thought was to use the vessels schedules to solve this issue, but quickly discovered that the schedules were only available for a subset of the vessels present in the ais test dataset.

By looking at public guidelines regarding AIS data, one can see that there are valid and invalid intervals and values of data for each feature included in ais_train. For SOG values, the expected valid values are in the range 0-102.2 knots. The feature has an error value of 1023 knots. For rot values, the expected valid values are –126 to 126 for basic actions. The feature also has an error of –128. The expected valid values for heading are between 0 and 359.9 degrees. The feature does not include 360 degrees as a valid value and has an additional error value at 511 degrees (*Class A AIS Position Report (Messages 1, 2, and 3) | Navigation Center*, n.d.).

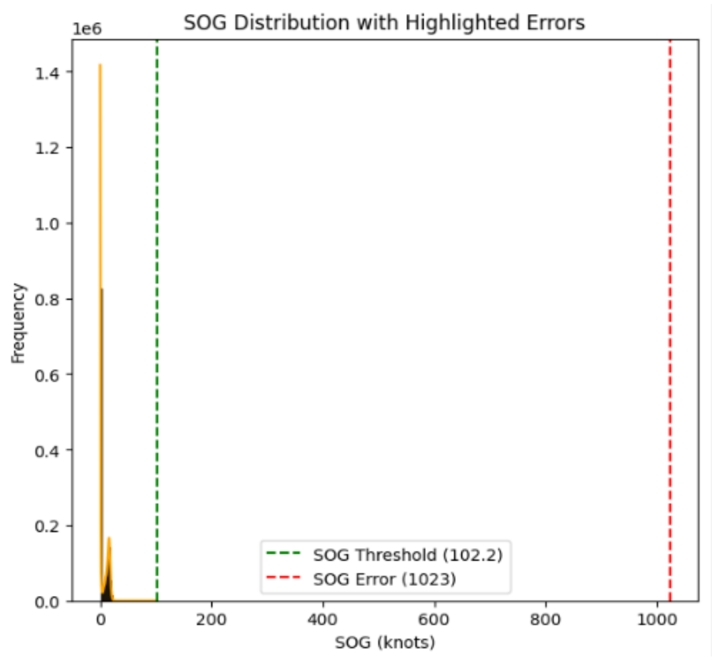
With this knowledge in mind, one can check the given intervals of values for the dataset. The following is a plot of the previously clarified intervals/singular error values:



Plot of heading values with threshold and errors



Plot of rot values with threshold and errors



Plot of sog values with threshold and errors (appendix 2 includes all three plots)

This results in 393 instances of invalid SOG values, 27944 instances of invalid ROT values and 4896 instances of invalid heading values. These results are crucial for the preprocessing step.

3. Data Preprocessing and Feature Engineering

Data cleaning and feature engineering are fundamental steps in machine learning, as they ensure the reliability and interpretability of models. Properly cleaned and engineered data helps avoid learning from flawed or misleading information and enhances model performance by capturing relevant patterns in the data. In this analysis, we explore the data preprocessing strategies used in two implementations: Short_notebook_1 and Short_notebook_2. Both notebooks utilize unique approaches to prepare Automatic Identification System (AIS) data for vessel tracking, with each focusing on specific methods for handling data inconsistencies, feature engineering, and model enhancement.

In Short_notebook_1, the data cleaning process begins with defining valid intervals for key features, including:

- COG (Course Over Ground) interval: [0, 360)
- SOG (Speed Over Ground) interval: [0, 102.2)
- ROT (Rate of Turn) interval: (-126, 126)
- Heading interval: [0, 360)
- Navstat (Navigational Status) interval: [0, 15]
- Latitude interval: (-90, 90)
- Longitude interval: (-180, 180)

Invalid values are identified based on these intervals and replaced with the mean of each respective column. Missing VesselId values are assigned the label "Unknown." These steps ensure data integrity and reduce the risk of outliers influencing the model. In addition, a mean imputer addresses any remaining NaN values before model training.

Short_notebook_2 follows a similar approach to Short_notebook_1 for defining valid intervals but introduces additional preprocessing steps to handle missing or incorrect data across multiple datasets, including ais_train.csv, ais_test.csv, vessels.csv and ports.csv. Columns irrelevant to the prediction task (e.g., UN_LOCODE, ISO, portLocation) are removed to streamline the dataset, while invalid port coordinates are filtered out. This thorough cleaning ensures that the model does not encounter unnecessary noise or inconsistencies.

Feature engineering is essential to improve model performance by enabling the model to capture spatial, temporal, and contextual information. Both Short_notebook_1 and Short_notebook_2 leverage feature engineering with a focus on lagged and differenced values, which are critical for tracking vessel movement over time.

In Short_notebook_1, lagged values for core features (e.g., cog, sog, rot, heading, navstat) provide information on a vessel’s previous state, helping the model understand transitions and patterns in its journey. Differenced values, such as latitude_diff and longitude_diff, capture directional changes in position. Additionally, the estimated_distance feature calculates distance based on the last-known speed and time difference, providing a projection of distance traveled. Although the feature distance_to_nearest_port was initially considered, it was excluded from the final model due to time constraints and limited impact on model accuracy.

Short_notebook_2 builds on the approach in Short_notebook_1 by implementing a broader set of features, such as interaction features (sog_heading_interaction, cog_rot_interaction, delta_time_distance_interaction) that capture complex relationships between variables. Furthermore, time-based features like hour_of_day, day_of_week, and is_weekend are introduced to allow the model to account for temporal patterns in vessel behavior. To capture spatial proximity, Short_notebook_2 also introduces a distance_to_nearest_port feature calculated using a BallTree for efficient geospatial nearest-neighbor search. This feature enriches the model with contextual information about the vessel’s location relative to ports.

Both notebooks utilize ensemble models to improve predictive performance, with Short_notebook_2 implementing a range of algorithms, including RandomForest, XGBoost, ElasticNet, LightGBM, and CatBoost. To combine the predictions from these models effectively, Short_notebook_2 introduces a combined loss function that integrates Mean Absolute Error (MAE) with Haversine distance, optimized through an ensemble weighting strategy.

4. Model Development

Effective model selection and training are essential in developing a predictive model that balances accuracy, interpretability, and computational feasibility. For this task of vessel positioning prediction, several approaches were explored, aiming to capture the time-dependent and sequential nature of Automatic Identification System (AIS) data while adhering to computational constraints. The following section discusses the models considered, training processes, evaluation criteria, and final selection for Short_notebook_1.ipynb and Short_notebook_2.ipynb with some variations and additions.

A diverse set of models was tested to manage the complexity and structure of AIS data, which includes time series with irregular intervals, missing values, and multi-dimensional output (latitude and longitude). The selection of models reflects an intent to strike a balance between accurately predicting vessel positions and managing computational demands within a 12-hour limit on standard computing resources.

Long Short-Term Memory (LSTM): The LSTM model was initially selected due to its effectiveness in capturing long-term dependencies and sequential patterns in time series data. LSTM models are specifically designed to learn from ordered data, making them suitable for tracking vessel movements over time. In this approach, vessel data was grouped by vessel ID, and sequences were created to capture trends over varying time spans (e.g., 5-day and 30-day sequences). However, LSTM's predictive power was hindered by high error scores, potentially because it was configured to predict one position per day for each vessel, which did not account for the variations in hour and minute timestamps. Additionally, the model struggled with longer-term predictions, likely due to a loss of broader route patterns across vessels. Given the computational intensity and the suboptimal results, the LSTM model was determined to be impractical within the given constraints. (GeeksforGeeks, 2024)

Regression Models: The primary focus shifted to regression-based models due to their efficiency and adaptability with structured data. XGBoost and Random Forest were selected as candidates for their capabilities in modeling non-linear relationships in time series data, handling missing values, and managing irregular intervals. Regression models also tend to be less computationally intensive, making them well-suited to the time restrictions of this project

In Short_notebook_1.ipynb, XGBoost emerged as the primary model due to its robustness in managing time-based data with gaps and inconsistencies. XGBoost is particularly well-suited for modeling non-linear relationships, which can be crucial for predicting complex patterns in vessel movements. The model was configured within a MultiOutputRegressor framework, allowing it to predict both latitude and longitude simultaneously. This setup ensures that the model can account for the multi-dimensional nature of the target variable while handling missing data efficiently through XGBoost's internal handling capabilities. (GeeksforGeeks, 2023)

The training process involved a SimpleImputer with a mean strategy to fill missing values, ensuring data completeness and stability. Key hyperparameters, such as n_estimators (set to 200), reg_lambda, and alpha, were optimized based on extensive testing to achieve a balance between accuracy and overfitting. The evaluation metrics included Mean Absolute Error (MAE) and Mean Squared Error (MSE), which provided a comprehensive assessment of the model's predictive accuracy and consistency. The training process incorporated cross-validation with 5 folds to ensure robust performance across different data splits, thereby reducing the risk of overfitting to specific subsets of the data.

Random Forest was also evaluated for this task as it is known for its efficiency with structured data and offers relatively high interpretability. However, Random Forest's limited regularization options and reduced flexibility in adapting to complex time-based patterns made it less suited for this problem compared to XGBoost. Although Random Forest achieved moderate performance, XGBoost provided a higher degree of control over the model's structure and hyperparameters, making it the preferred model in Short_notebook_1.ipynb.

In Short_notebook_2.ipynb, the model development involved testing and improving different regression algorithms, all used within a MultiOutputRegressor setup to predict latitude and longitude at the same time. The goal was to find the best model for accurately predicting vessel positions while keeping computational costs manageable.

The evaluation involved a variety of regression models, each selected for their unique potential in managing temporal data and multi-output prediction tasks. Hyperparameter tuning was a critical component, aimed at enhancing each model's capacity for generalization and its predictive precision. Additionally, the selected models were analyzed based on their computational efficiency, a necessary consideration for scalability and adaptability across diverse vessel trajectories and timestamps. This rigorous approach allowed Short_notebook_2 to identify the optimal regression model for achieving reliable and accurate vessel movement predictions.

The following table provides a summary of the performance of each model tested across the two notebooks:

Model	**MAE**	**MSE**	**Efficiency**
LSTM	High	High	Computationally intensive, high resource demand
Random forest	Moderate	Moderate	Efficient but limited in handling complex
XGBoost	Low	Low	Effective, robust handling of missing values, strong generalization

The final models chosen for Short_notebook_1.ipynb and Short_notebook_2.ipynb were both based on XGBoost due to its robust performance and adaptability to structured AIS data:

Short_notebook_1: The final model was XGBoost with MultiOutputRegressor. This model was selected because it could manage missing values, capture non-linear relationships, and handle the multi-dimensional nature of the target variable efficiently. XGBoost's configurability allowed for fine-tuning to prevent overfitting, yielding a strong predictive model that balanced accuracy with computational feasibility.

Short_notebook_2: XGBoost was also chosen as the final model, with an expanded feature set that included interaction terms and temporal features. This enhanced feature set allowed the model to capture complex dependencies and spatial patterns, such as proximity to ports, which are important for predicting vessel trajectories. The additional features in Short_notebook_2 provided the model with a deeper contextual understanding, enabling it to generalize more effectively to new data.

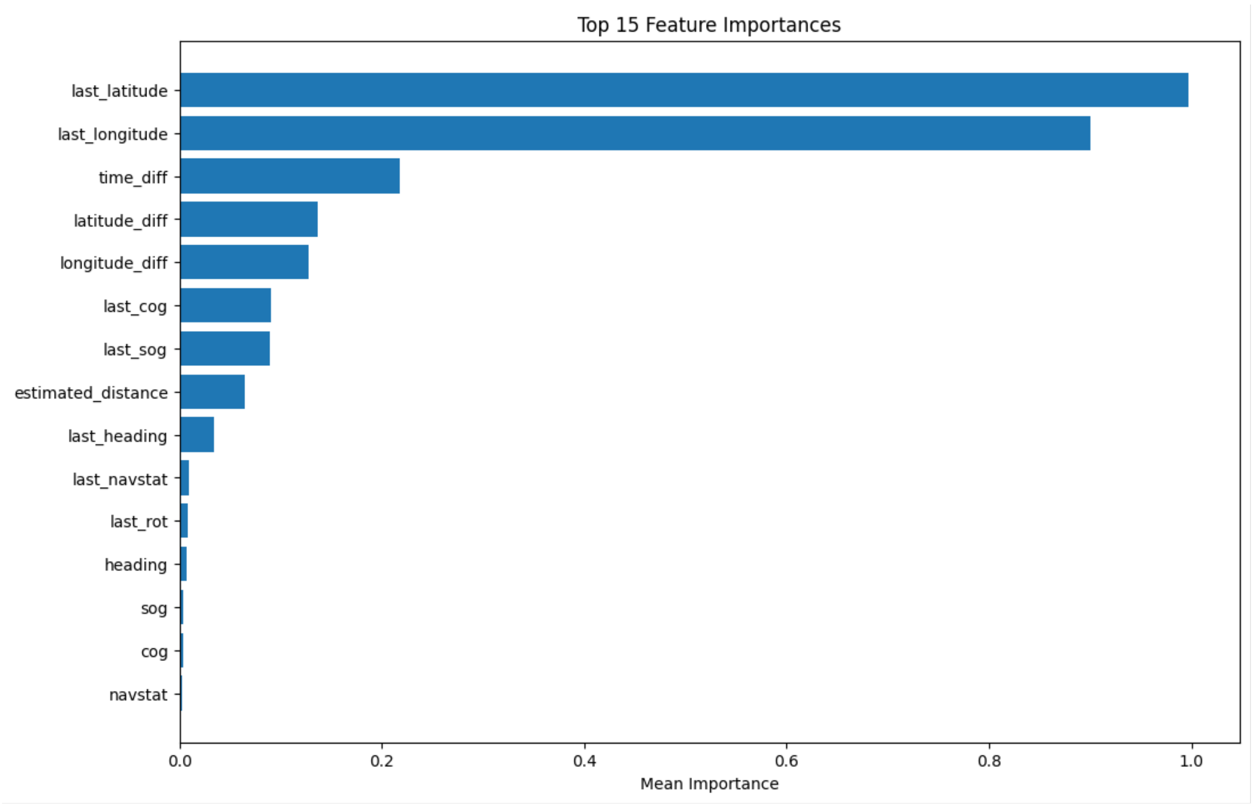
In conclusion, while the LSTM model offered theoretical advantages in capturing sequential dependencies, the computational constraints and training time limitations led to the selection of

XGBoost as the optimal solution for both notebooks. By balancing performance, interpretability, and computational efficiency, XGBoost provided a practical, high-performing model that successfully addressed the complexities of AIS data within the project's constraints.

5. Model Interpretation

Model interpretation is a critical aspect of machine learning, so we can understand how the model arrives at its predictions. This section delves into the analysis of feature importance, model explainability, and the examination of model errors to provide a comprehensive understanding of the XGBoost models developed in both Short_notebook_1.ipynb and Short_notebook_2.ipynb.

Understanding which features significantly influence the model's predictions is essential for both model transparency and potential feature engineering. Feature importance analysis was conducted using permutation importance and SHAP (SHapley Additive exPlanations) values, providing complementary perspectives on feature contributions for both notebooks.



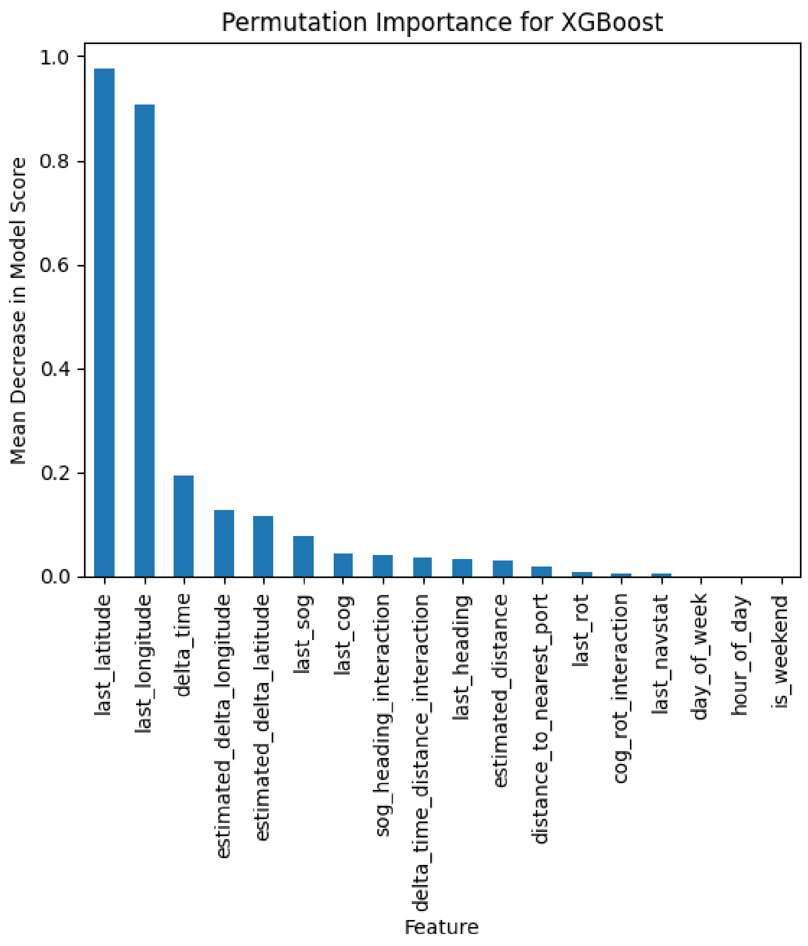
Plot of the permutation feature importance for the specified XGBoost model in Short_notebook_1.ipynb (Appendix 3)

The permutation feature importance plot for Short_notebook_1.ipynb highlights the most influential features in predicting vessel latitude and longitude. This shows that last_latitude and last_longitude show the highest importance, underscoring their role as strong predictors of future vessel positions. This is intuitive, as the most recent coordinates provide direct information about the vessel’s current location.

Other significant features include:

- time_diff: Reflects the time interval between consecutive data points, crucial for estimating the distance a vessel might travel.

- latitude_diff and longitude_diff: Capture recent changes in position, aiding in understanding the vessel's trajectory.
- last_cog (Course Over Ground) and last_sog (Speed Over Ground): Provide context about the vessel's previous direction and speed, informing future movement predictions.
- estimated_distance: Estimates the distance traveled based on the last known speed and elapsed time, further supporting position predictions.
- last_heading: Offers additional directional information, enhancing the model's ability to forecast vessel paths.
- last_navstat (Navigational Status) and last_rot (Rate of Turn): Although ranked lower, these features provide supplementary data on navigation status and vessel maneuvering, influencing positional changes.



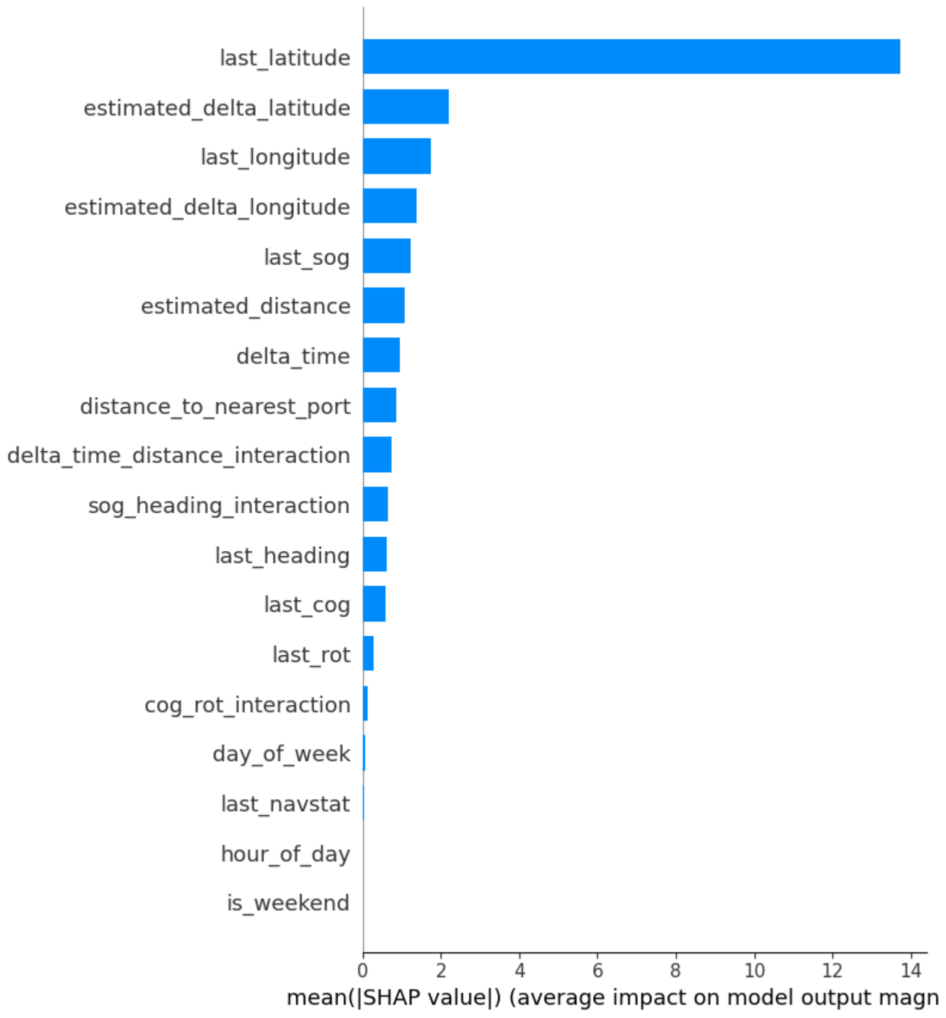
Permutation Feature Importance for XGBoost Model in Short_notebook_2.ipynb (Appendix 5)

In Short_notebook_2.ipynb, the feature importance analysis expands upon Short_notebook_1 by incorporating additional engineered features. The permutation feature importance plot reveals that, in addition to last_latitude and last_longitude, interaction features such as sog_heading_interaction and cog_rot_interaction play significant roles.

Key additional features include:

- sog_heading_interaction: Captures the combined effect of speed and heading, providing nuanced insights into vessel movement.
- cog_rot_interaction: Reflects the interplay between course over ground and rate of turn, aiding in predicting directional changes.
- hour_of_day and day_of_week: Incorporate temporal patterns, allowing the model to account for time-based behaviors in vessel movements.
- distance_to_nearest_port: Although initially excluded in Short_notebook_1, this feature is included in Short_notebook_2 to enrich spatial context, calculated using a BallTree for efficient geospatial nearest-neighbor search.

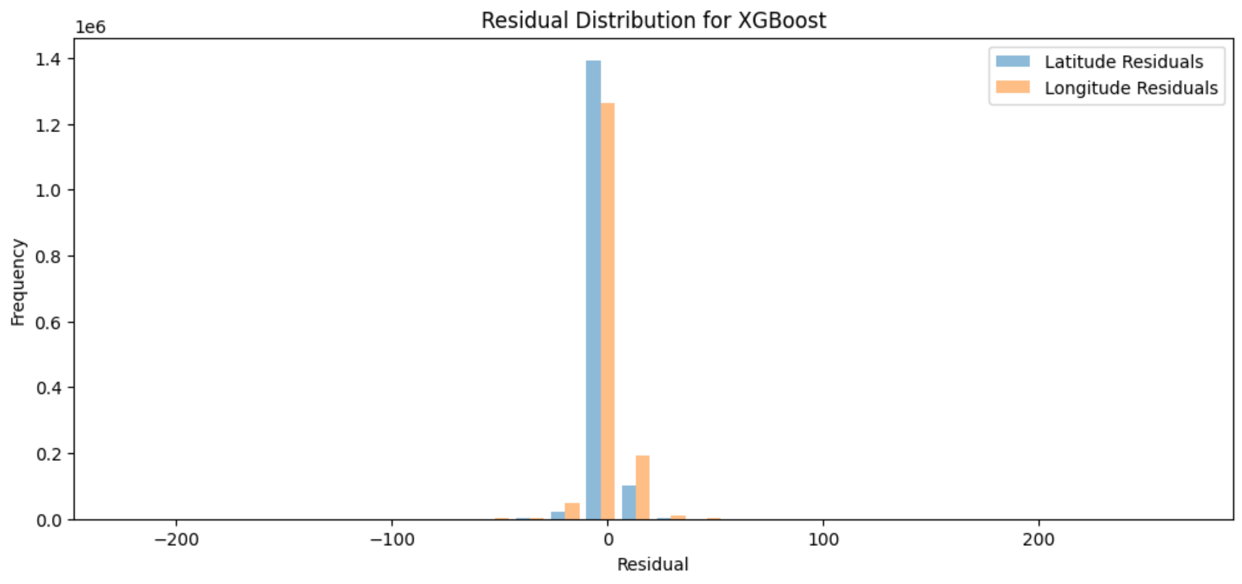
Beyond feature importance, model explainability tools like SHAP were employed to clarify the decision-making process of the XGBoost models.



SHAP Summary Plot for XGBoost Model in Short_notebook_2.ipynb (Appendix 5)

SHAP values provide a unified measure of feature importance by quantifying each feature's contribution to individual predictions.

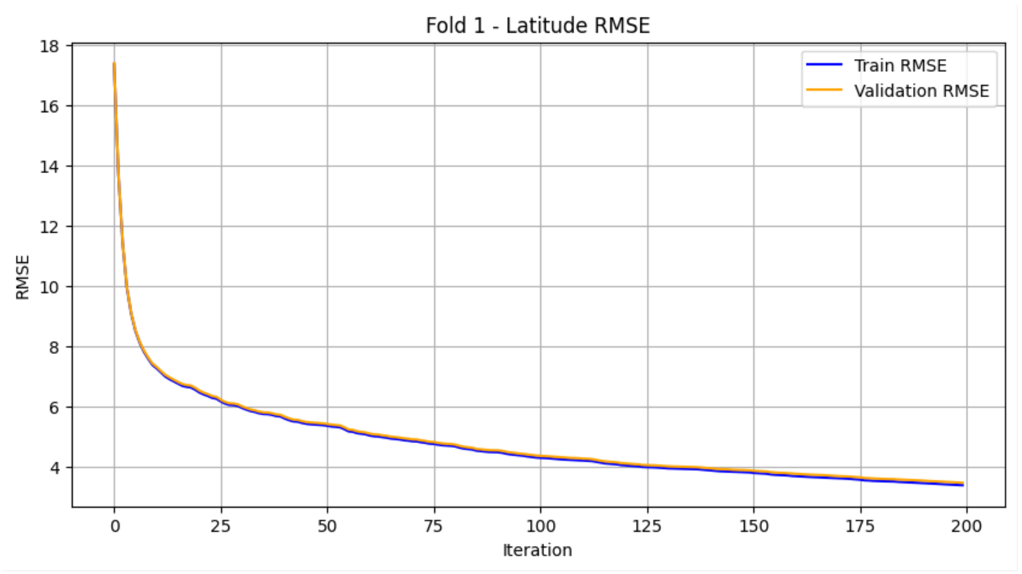
Analyzing where and why the model underperforms is crucial for enhancing its robustness and reliability. Residual analysis and error distribution examination were conducted to identify patterns and potential areas of improvement in both notebooks.



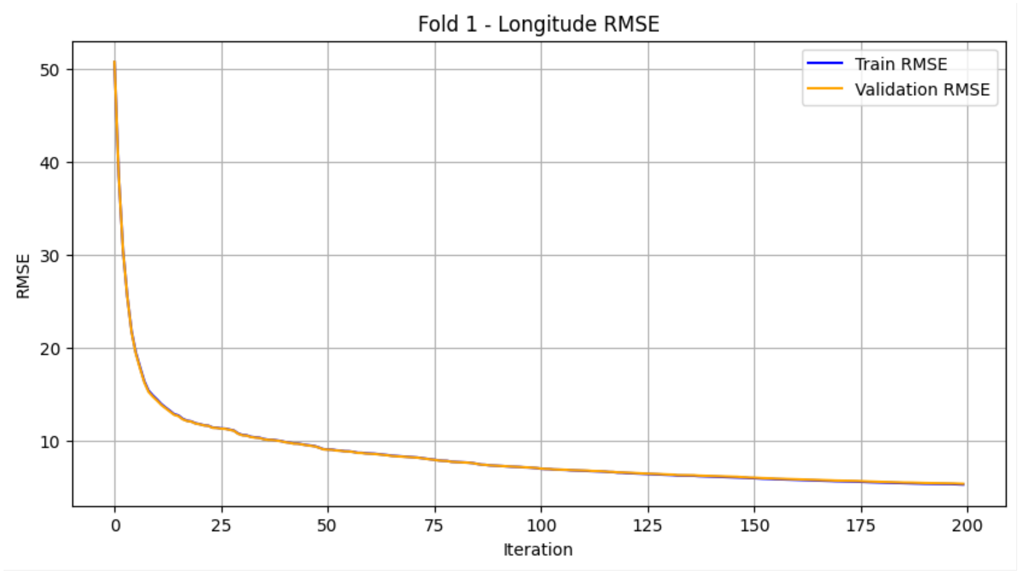
Residuals in Short_notebook_2.ipynb (Appendix 5)

The residual distribution of the XGBoost model used for predicting latitude and longitude positions demonstrates a high degree of accuracy, as evidenced by the concentration of residuals around zero. This narrow distribution, centered at zero, indicates that the model's predictions closely approximate the actual values for both latitude and longitude, suggesting low error rates across the dataset. Additionally, the symmetrical nature of the distribution, with few outliers, implies that the model maintains a balanced performance in both dimensions of prediction, effectively minimizing bias in either direction. The few larger residuals—appearing as outliers—highlight areas where the model's predictions diverge more significantly.

Plot of latitude and longitude root-mean-square-error (RMSE) in fold 1:



Plot of the latitude RMSE in the first fold for the specified XGBoost model in Short_notebook_1.ipynb (Appendix 4)



Plot of the longitude RMSE in the first fold for the specified XGBoost model in Short_notebook_1.ipynb (Appendix 4)

Inspecting the graph, the training and validation loss curves show a steady decrease, eventually flattening out as the number of iterations increases. This pattern suggests that the model is converging effectively toward an optimal state. Notably, the training loss remains only slightly below the validation loss throughout, fulfilling the expectation of good generalization without overfitting. The parallel movement of the two curves without significant divergence confirms a balanced learning process, where the model shows consistent improvement on both the training and validation data. This indicates that the model is achieving an appropriate fit for the data, minimizing the risk of overfitting or underfitting while maintaining effective learning throughout the training.

Overall, the XGBoost models in both notebooks demonstrate strong predictive capabilities, with feature importance and SHAP analysis providing valuable insights into the driving factors behind predictions. The low residuals and balanced RMSE across training and validation sets indicate robust model performance and effective generalization to new data.

6. Conclusion and future work

This project successfully addressed the challenge of predicting future vessel coordinates using machine learning methods applied to Automatic Identification System (AIS) data. By leveraging exploratory data analysis, data preprocessing, and feature engineering, we developed predictive models capable of forecasting vessel positions from May 8th to May 12th based on historical data from January 1st to May 7th. The primary modeling approach centered on XGBoost within a MultiOutputRegressor framework.

Key insights from feature importance and SHAP (SHapley Additive exPlanations) analyses revealed that recent positional data (e.g., last_latitude, last_longitude), time intervals between data points (time_diff), and dynamic movement features (e.g., latitude_diff, longitude_diff, last_cog, last_sog) were critical in accurately predicting future vessel locations. Additionally, the incorporation of interaction in Short_notebook_2.

The final models shows strong generalization capabilities by low residuals and balanced RMSE across training and validation datasets. This indicates that the models minimized overfitting and maintained consistent predictive accuracy across diverse vessel trajectories and temporal contexts.

Data Quality and Availability: The AIS datasets presented significant challenges, including irregularly spaced timestamps, missing values, and a substantial number of vessels with limited data. These issues potentially constrained the models' ability to generalize across all vessel trajectories.

Feature Engineering Constraints: While feature engineering was performed, certain potentially valuable features, such as distance_to_nearest_port, were excluded in some models due to time constraints and limited initial impact on model performance. This may have restricted the models' ability to fully leverage spatial contextual information.

Model Complexity and Computational Resources: Advanced deep learning models like LSTM were deemed impractical within the project's computational and temporal constraints. Consequently, the study primarily focused on regression-based models, which, although effective, may not capture all sequential dependencies present in the data.

Building upon the current findings, several avenues for future research and development are proposed to enhance predictive accuracy and model robustness:

Hybrid Modeling Approaches: Integrating deep learning techniques, such as LSTM, with traditional regression models could capitalize on the strengths of both methodologies. For instance, LSTM models can effectively capture the sequential nature of vessel movements on a per-vessel basis, while regression models like XGBoost can identify broader trends and patterns across the entire dataset. This hybrid approach may yield more accurate predictions by addressing both vessel-specific dynamics and global movement patterns.

Advanced Deep Learning Architectures: Exploring transformer-based models, such as the Temporal Fusion Transformer (TFT), could further enhance the ability to model complex temporal dependencies and interactions across multiple vessels. Transformers' attention mechanisms can potentially uncover relationships and shared routes among vessels, improving the prediction of coordinated or correlated movements.

Automated Machine Learning (AutoML): Implementing AutoML frameworks can streamline the process of model selection and hyperparameter tuning, thereby increasing efficiency and potentially identifying more optimal model configurations. AutoML can help test more algorithms and settings, improving predictive performance without much manual effort.

Enhanced Feature Engineering: Expanding the feature set to include more sophisticated interaction terms and temporal features can provide deeper insights into vessel behavior. Incorporating advanced geospatial features, such as dynamic distance metrics to multiple ports or maritime landmarks and integrating external data sources like weather conditions or sea traffic density could further enrich the model's contextual understanding and predictive capabilities.

In summary, while the current study lays a foundation for vessel position prediction using machine learning, the proposed future improvements offer pathways to advance the methodology, address existing limitations, and achieve higher levels of accuracy.

7. References

Class A AIS Position Report (Messages 1, 2, and 3) | Navigation Center. (n.d.). https://www.navcen.uscg.gov/ais-class-a-reports

GeeksforGeeks. (2023, February 6). *XGBoost*. GeeksforGeeks. https://www.geeksforgeeks.org/xgboost/

GeeksforGeeks. (2024, June 10). *What is LSTM Long Short Term Memory?* GeeksforGeeks.

https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/

8. Appendixes

Appendix 1: Plotting vessels

```
import matplotlib.pyplot as plt


# Filter train_df to include only vesselIds present in ais_test
train_df_in_test = train_df[train_df["vesselId"].isin(test_df["vesselId"])]


# Extract year and month from the time column
train_df_in_test["year_month"] = train_df_in_test["time"].dt.to_period("M")


# Group by vesselId and year_month to count occurrences
occurrences_per_month = (
train_df_in_test.groupby(["vesselId", "year_month"]).size().unstack(fill_value=0)
)

# Filter to include only vesselIds with less than 10 datapoints in May
vessels_less_than_10_in_may = occurrences_per_month[
occurrences_per_month[
occurrences_per_month.columns[occurrences_per_month.columns.month == 5]
].sum(axis=1)
< 10
]

# Plot the occurrences per vesselId per month for filtered vessels
plt.figure(figsize=(15, 10))
vessels_less_than_10_in_may.T.plot(
kind="bar", stacked=True, figsize=(15, 10), legend=False
)
plt.title(
"Occurrences of Data Points in ais train per VesselId per Month (VesselIds in ais_test with < 10 datapoints in
May)"
)
plt.xlabel("Month")
plt.ylabel("Number of Data Points")
plt.legend(title="VesselId", bbox_to_anchor=(1.05, 1), loc="upper left")
plt.tight_layout()
plt.show()
```

Appendix 2: Plotting features and errors

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Load the AIS train data with straight slash as delimiter
ais_train = pd.read_csv('ais_train.csv', sep='|' )

# Convert 'time' column to datetime format
ais_train['time'] = pd.to_datetime(ais_train['time'], errors='coerce')

# Function to detect specific outliers and count their occurrences
def detect_specific_outliers(df):
    conditions = {
        'sog_error': df[(df['sog'] == 1023) | (df['sog'] > 102.2)],
        'rot_error': df[(df['rot'] == 128) | (df['rot'] < -126) | (df['rot'] > 126)],
        'heading_error': df[(df['heading'] < 0) | (df['heading'] > 359) | (df['heading']==511)],
    }
    counts = {key: len(value) for key, value in conditions.items()}
    value_counts = {key: value.iloc[:, 0].value_counts() for key, value in conditions.items()}
    return counts, value_counts

# Identify specific outliers
outlier_counts, outlier_value_counts = detect_specific_outliers(ais_train)

# Visualize COG, SOG, and ROT distributions with specific conditions highlighted
plt.figure(figsize=(18, 6))



# Plot SOG distribution
plt.subplot(1, 3, 2)
sns.histplot(ais_train['sog'], kde=True, color='orange', bins=50)
plt.axvline(x=102.2, color='green', linestyle='--', label='SOG Threshold (102.2)')
plt.axvline(x=1023, color='red', linestyle='--', label='SOG Error (1023)')
plt.title('SOG Distribution with Highlighted Errors')
plt.xlabel('SOG (knots)')
plt.ylabel('Frequency')
plt.legend()

# Plot ROT distribution
```

```
plt.subplot(1, 3, 3)
sns.histplot(ais_train['rot'], kde=True, color='green', bins=50)
plt.axvline(x=-126, color='blue', linestyle='--', label='ROT Lower Bound (-126)')
plt.axvline(x=126, color='blue', linestyle='--', label='ROT Upper Bound (126)')
plt.axvline(x=128, color='red', linestyle='--', label='ROT Error (128)')
plt.title('ROT Distribution with Highlighted Error and Boundaries')
plt.xlabel('ROT')
plt.ylabel('Frequency')
plt.legend()

# Plot Heading distribution
plt.subplot(1, 3, 1)
sns.histplot(ais_train['heading'], kde=True, color='blue', bins=50)
plt.axvline(x=0, color='red', linestyle='--', label='Heading Lower Bound (0)')
plt.axvline(x=359, color='red', linestyle='--', label='Heading Upper Bound (359)')
plt.axvline(x=511, color='red', linestyle='--', label='Heading Error (511)')
plt.title('Heading Distribution with Highlighted Error and Boundaries')
plt.xlabel('Heading')
plt.ylabel('Frequency')
plt.legend()




plt.tight_layout()
plt.show()

# Print outlier counts and occurrence details
print("Outlier counts for specific conditions:")
for key, count in outlier_counts.items():
    print(f"{key}: {count} instances")

# Display occurrences of specific outlier values
print("\nOccurrences of specific outlier values:")
for key, value_counts in outlier_value_counts.items():
    if not value_counts.empty:
        print(f"\n{key}:")
        print(value_counts.head())
```

Appendix 3: Permutation feature importance

```
# Permutation importance
from sklearn.inspection import permutation_importance
from sklearn.multioutput import MultiOutputRegressor
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
import xgboost
import matplotlib.pyplot as plt

# Load and prepare your data as in your original code
training_data = pd.read_csv('final_train.csv')
input_features = training_data.drop(columns=['latitude', 'longitude', 'vesselId', 'time'])
target_outputs = training_data[['latitude', 'longitude']]
X_test = test_data[input_features.columns]

# Preprocess categorical columns
cat_columns = input_features.select_dtypes(include=['object', 'category']).columns
for column in cat_columns:
    input_features[column] = input_features[column].astype('category')
    test_data[column] = test_data[column].astype('category')

# Downcast numerical columns
for column in input_features.select_dtypes(include=['float64']).columns:
    input_features[column] = pd.to_numeric(input_features[column], downcast='float')

# Set up the model
xgb_model = MultiOutputRegressor(
    Pipeline([
        ('fill_na', SimpleImputer(strategy='mean')),
        ('xgb_model', xgboost.XGBRegressor(
            objective='reg:squarederror',
            n_estimators=200,
            seed=42,
            tree_method='gpu_hist',
            gpu_id=0,
            verbosity=0,
            reg_lambda=1.0,
            alpha=0.1,
        ))
    ])
)

# Fit the model
xgb_model.fit(input_features, target_outputs)

# Calculate permutation importance separately for latitude and longitude
perm_importance_lat = permutation_importance(
    xgb_model.estimators_[0], input_features, target_outputs['latitude'], n_repeats=5, random_state=42
)

perm_importance_lon = permutation_importance(
    xgb_model.estimators_[1], input_features, target_outputs['longitude'], n_repeats=5, random_state=42
)

# Create DataFrame for importance scores
importance_df = pd.DataFrame({
    'Feature': input_features.columns,
    'Importance_Latitude': perm_importance_lat.importances_mean,
    'Importance_Longitude': perm_importance_lon.importances_mean,
    'Mean_Importance': (perm_importance_lat.importances_mean + perm_importance_lon.importances_mean) /
2
})

# Sort by mean importance
importance_df = importance_df.sort_values('Mean_Importance', ascending=False)
```

```
# Print top 15 features
print("\nTop 15 Most Important Features:")
print(importance_df[['Feature', 'Mean_Importance']].head(15))

# Plot feature importance
plt.figure(figsize=(12, 8))
plt.barh(importance_df['Feature'].head(15), importance_df['Mean_Importance'].head(15))
plt.gca().invert_yaxis()
plt.xlabel('Mean Importance')
plt.title('Top 15 Feature Importances')
plt.show()
```

Appendix 4: Latitude and longitude RMSE

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.impute import SimpleImputer
import xgboost as xgb
from sklearn.model_selection import KFold

# Load data
test_data = pd.read_csv('final_test.csv')
training_data = pd.read_csv('final_train.csv')

# Split features and target columns
input_features = training_data.drop(columns=['latitude', 'longitude', 'vesselId', 'time'])
target_outputs = training_data[['latitude', 'longitude']]
```

```
# Ensure test_data has all required columns
for column in input_features.columns:
    if column not in test_data.columns:
        test_data[column] = np.nan

X_test = test_data[input_features.columns]

# Impute missing values
imputer = SimpleImputer(strategy='mean')
input_features = imputer.fit_transform(input_features)
X_test = imputer.transform(X_test)

# Cross-validation setup
num_folds = 5
cross_validator = KFold(n_splits=num_folds, shuffle=True, random_state=42)
split_indices = list(cross_validator.split(input_features, target_outputs))

# Function for training and plotting XGBRegressor models
def run_model(X_train, y_train, X_validate, y_validate, fold_idx, target_name):
    model = xgb.XGBRegressor(
        objective='reg:squarederror',
        n_estimators=200,
        seed=42,
        tree_method='gpu_hist',
        gpu_id=0,
        verbosity=1,
        reg_lambda=1.0,
        alpha=0.1
    )

    # Fit the model with a validation set to track performance
    model.fit(X_train, y_train, eval_set=[(X_train, y_train), (X_validate, y_validate)], verbose=False)

    # Extract and plot evaluation metrics
    evals_result = model.evals_result()
    plt.figure(figsize=(10, 5))
    plt.plot(evals_result['validation_0']['rmse'], label='Train RMSE', color='blue')
    plt.plot(evals_result['validation_1']['rmse'], label='Validation RMSE', color='orange')
    plt.title(f'Fold {fold_idx + 1} - {target_name} RMSE')
    plt.xlabel('Iteration')
    plt.ylabel('RMSE')
    plt.legend()
    plt.grid(True)
    plt.show()

    return model

# Train separate models for each target
models = {}
for fold_idx, (train_index, validate_index) in enumerate(split_indices):
    X_train, X_validate = input_features[train_index], input_features[validate_index]
    y_train_lat, y_validate_lat = target_outputs['latitude'].iloc[train_index],
target_outputs['latitude'].iloc[validate_index]
    y_train_lon, y_validate_lon = target_outputs['longitude'].iloc[train_index],
target_outputs['longitude'].iloc[validate_index]

    # Train and plot for latitude
    model_lat = run_model(X_train, y_train_lat, X_validate, y_validate_lat, fold_idx, 'Latitude')
    models[f'fold_{fold_idx}_latitude'] = model_lat

    # Train and plot for longitude
    model_lon = run_model(X_train, y_train_lon, X_validate, y_validate_lon, fold_idx, 'Longitude')
    models[f'fold_{fold_idx}_longitude'] = model_lon
```

Appendix 5: Feature importance

```
import shap
from sklearn.inspection import permutation_importance, PartialDependenceDisplay

sample_fraction = 0.2
X_sample = X_train.sample(frac=sample_fraction, random_state=42)
y_sample = y_train.loc[X_sample.index]

for model_name, multi_output_model in models.items():
if hasattr(multi_output_model, 'estimators_'):
importances = []
for est in multi_output_model.estimators_:
if hasattr(est, 'feature_importances_'):
importances.append(est.feature_importances_)
if importances:
avg_importances = np.mean(importances, axis=0)
feature_importance_df = pd.DataFrame({
'Feature': feature_columns,
'Importance': avg_importances
}).sort_values(by='Importance', ascending=False)

plt.figure(figsize=(10, 6))
plt.title(f"Aggregated Feature Importances for {model_name}")
plt.barh(feature_importance_df['Feature'], feature_importance_df['Importance'])
plt.xlabel("Importance")
plt.gca().invert_yaxis()
plt.show()

for model_name, multi_output_model in models.items():
if hasattr(multi_output_model, 'estimators_'):
importances = []
for i, estimator in enumerate(multi_output_model.estimators_):
if hasattr(estimator, 'predict'):
perm_importance = permutation_importance(
estimator, X_sample, y_sample.iloc[:, i], n_repeats=5, random_state=42
)
importances.append(perm_importance.importances_mean)

if importances:
aggregated_importance = np.mean(importances, axis=0)
perm_importance_df = pd.DataFrame({
'Feature': feature_columns,
'Importance': aggregated_importance
}).sort_values(by='Importance', ascending=False)
```

```
plt.figure(figsize=(10, 6))
perm_importance_df.plot(kind='bar', x='Feature', y='Importance', legend=False)
plt.title(f"Permutation Importance for {model_name}")
plt.ylabel('Mean Decrease in Model Score')
plt.show()

for model_name, multi_output_model in models.items():
if 'LightGBM' in model_name or 'XGBoost' in model_name:
selected_estimator = multi_output_model.estimators_[0]
explainer = shap.TreeExplainer(selected_estimator)
shap_values = explainer.shap_values(X_train)
print(f"SHAP Summary Plot for {model_name}")
shap.summary_plot(shap_values, X_train, plot_type="bar")

if 'RandomForest' in models:
rf_model = models['RandomForest'].estimators_[0]

fig, ax = plt.subplots(figsize=(8, 6))
PartialDependenceDisplay.from_estimator(rf_model, X_train, [0, 1], ax=ax)
plt.show()

for model_name, preds in oof_preds.items():
residuals = y_train.values - preds
residual_mean = residuals.mean(axis=0)
residual_std = residuals.std(axis=0)
print(f"{model_name} Residual Mean (Latitude, Longitude): {residual_mean}")
print(f"{model_name} Residual Std (Latitude, Longitude): {residual_std}")

plt.figure(figsize=(12, 5))
plt.hist(residuals, bins=30, label=['Latitude Residuals', 'Longitude Residuals'], alpha=0.5)
plt.title(f"Residual Distribution for {model_name}")
plt.xlabel("Residual")
plt.ylabel("Frequency")
plt.legend()
plt.show()
```