Mini Project 1 Report

Maša Ćirković

I. Introduction

With fuel efficiency being a critical factor in maritime transportation, optimizing fuel consumption is essential for reducing operational costs and minimizing environmental impact. Accurately predicting fuel consumption can enable better route planning, usage of wind speed and angle, maintenance scheduling, and overall energy efficiency in shipping operations.

This report focuses on developing a machine learning model to predict fuel consumption for ships based on various operational and environmental parameters. The dataset consists of multiple sensor readings, including fuel density, volume flow rate, speed, wind conditions, heading, and water speed, recorded at high-frequency intervals. The data has been aggregated into hourly windows to capture meaningful trends while maintaining computational efficiency, and the predictions are hourly-based as well.

Different machine learning approaches are explored, including Random Forest, XGBoost, and Linear regressors, to determine which model provides the most accurate and reliable predictions. Additionally, feature engineering techniques, such as combining wind speed and angle, have been applied to enhance model performance.

By improving fuel consumption predictions, this analysis aims to contribute to more efficient maritime operations, helping shipping companies reduce costs and meet sustainability goals while maintaining optimal vessel performance.

II. Data Processing

Data Processing involves understanding what the data represents, exploring different data properties and transforming it into a suitable representation for further analysis or training the models.

A. Data Explanation

The dataset contained information about a Danish ro-ro passenger ship (MS Smyril). The MS Smyril has a length of 135m, a ship width of 22.7m, a design draft of 5.6m, and four 3,360kW main engines. The system collects data from the following sensor devices: the Doppler speed log, gyrocompass, Global Positioning System (GPS), main energy pipe flow meter, rudder angles, wind, propeller pitches, inclinometer, and level measurement device. The dataset was collected from February to April 2010, with a total of 246 voyages and 1,627,324 data records. The features are all saved in separate CSV files, which contain a timestamp (id), and the value for feature. Features are given in the table I.

Feature	Description
fuelDensity	Fuel density (kg/l)
fuelTemp	Fuel temperature (°C)
fuelVolumeFlowRate	Fuel volume flow rate (L/s)
inclinometer-raw	Inclinometer trim angle (degrees)
latitude	Latitude
longitude	Longitude
level1median	Port level measurements (m)
level2median	Starboard level measurements (m)
longitudinalWaterSpeed	Speed through water (STW) (kn)
portPitch	Port propeller pitch (-10 to 10 V)
portRudder	Port rudder angle (-10 to 10 V)
speedKmh	Speed over ground (SOG) (km/h)
speedKnots	Speed over ground (SOG) (kn)
starboardPitch	Starboard propeller pitch (-10 to 10 V)
starboardRudder	Starboard rudder angle (-10 to 10 V)
trackDegreeMagnetic	Track degree magnetic (degrees)
trackDegreeTrue	Track degree true (degrees)
trueHeading	True heading (degrees)
windAngle	Wind angle (degrees)
windSpeed	Wind speed (m/s)

TABLE I: *Description of features used in fuel consumption prediction*

B. Approach

Given that all the records are in separate files, they need to be joined based on their ID (timestamp). Since there is 1.6 million records, with their frequency being in milliseconds, it was decided to aggregate them on an hourly basis. This means that all of the data will be looked at in a span of 1 hour, and the predictions will also be hourly based.

The first step in the aggregation process was to convert the timestamp to a human-readable format, since it was given in the Windows .NET format. Initial thought was that it was a Unix timestamp, but the conversion gave the year 1990, which was not the same year as stated in the data set description. The code for converting it to a human readable format was the product of many llms working together, since Windows .NET is not the standard way of measuring time. After the timestamps were converted, library pandas has a built-in function to aggregate over time frames.

The next step was to determine how to aggregate numerical columns. For continuous real numerical values mean is used, and for angles circular mean was used. Latitude and longitude cannot be averaged in any of these ways, and thus it was explored to calculate total distance traveled in a window frame by summing all of the distances between adjacent points, but this idea was discarded given that we already have speed, so distance is not needed. Because of that, latitude and longitude were averaged the same as continuous real numerical values, but they will not be used for further calculations.

With all of this done, the 20 CSV files were merged together in one data frame based on the ID column. The type of join was outer, so that all of the records are included, regardless of

whether they have a corresponding pair. Rows with missing timestamps were dropped. There is **712** entries.

Fuel consumption column was added, and the formula for its calculation is for our basis and is given below.

$$\textbf{fuelConsumption} = \frac{\text{fuelDensity} \times \text{fuelVolumeFlowRate} \times 3600}{1000}$$

C. Feature Engineering

Feature engineering refers to creating new features, transforming existing ones, or selecting the most relevant features to improve model performance. In this project, a few of them were created.

- **Day of the week** was added because it made sense that fuel consumption is different from weekends to weekdays, and perhaps even between weekdays themselves.
- **Speed change** was also added and it was calculated as a change in speed between consecutive time points, because the rate of speed change can indicate the rate of fuel consumption.
- **Fuel flow rate change** was added in a similar manner to speed change, and it was calculated based on the fuel volume flow rate.
- **Lag 1 fuel** was added and it looked at the next time point for fuel consumption.
- **Lag 2 fuel** was added and it looked at the second next time point for fuel consumption.
- **Lag 1 speed** was added and it looked at the next time point for speed.
- **Effective wind speed** was calculated as a product of wind speed and the cosine of wind angle, giving insight into how wind affects ships speed.

The final dataframe had **28 columns** and **712 entries**.

Dimensionality of the data is high, and not every feature contributes to the fuel consumption as much. Before choosing features, correlation to the fuel consumption is explored and shown in the image 1.

fuelConsumption	1.000000	day_of_week	0.069259
fuelVolumeFlowRate	0.999899	inclinometer-raw	0.050448
portPitch	0.921816	windAngle	0.050242
speedKnots	0.901665	level1median	0.024690
speedKmh	0.901658	lag_1_fuel	-0.072748
longitudinalWaterSpeed	0.900046	lag_2_fuel	-0.126301
starboardPitch	0.894849	trackDegreeTrue	-0.133029
speed_change	0.728745	trackDegreeMagnetic	-0.162281
fuel_flow_rate_change	0.722880	lag_1_speed	-0.221389
portRudder	0.699394	latitude	-0.315717
level2median	0.587569	trueHeading	-0.327321
fuelTemp	0.436551	fuelDensity	-0.331975
windSpeed	0.362682	longitude	-0.482275
effective_wind_speed	0.349815	starboardRudder	-0.664518

Fig. 1: *Correlation of features in regards to the fuel consumption before removing features*

Based on this, a threshold was defined as $|0.2|$, meaning all features with correlation > -0.2 or < 0.2 will be removed.

- Features removed due to low importance:
  - *level1median*
  - *day_of_week*
  - *inclinometer-raw*
  - *lag_1_fuel*
  - *lag_2_fuel*
  - *trackDegreeTrue*
  - *trackDegreeMagnetic*
- Features removed due to high correlation with other features:
  - *speedKnots* (correlated with *speedKmh*)
  - *longitudinalWaterSpeed* (correlated with *speedKmh*)
  - *longitude* and *latitude* (already have speed, so distance is not needed)
  - *fuelVolumeFlowRate* (used to calculate *fuelConsumption*)
  - *fuelDensity* (used to calculate *fuelConsumption*)
  - *windSpeed* and *windAngle* (used to calculate *effective_wind_speed*)

After this, dataframe was left with **13 columns**.

Correlation is calculated once again for the remaining features and given in the table II.

Feature	Correlation
fuelConsumption	1.000000
portPitch	0.921816
speedKmh	0.901658
starboardPitch	0.894849
speed_change	0.728745
fuel_flow_rate_change	0.722880
portRudder	0.699394
level2median	0.587569
fuelTemp	0.436551
effective_wind_speed	0.349815
lag_1_speed	-0.221389
trueHeading	-0.327321
starboardRudder	-0.664518

TABLE II: *Correlation with fuelConsumption after removing features*

Correlation heatmap is a visual way of displaying correlations. Darker red colors indicate higher positive correlation, while darker blue indicate higher negative correlation. It can be viewed in the figure 2.

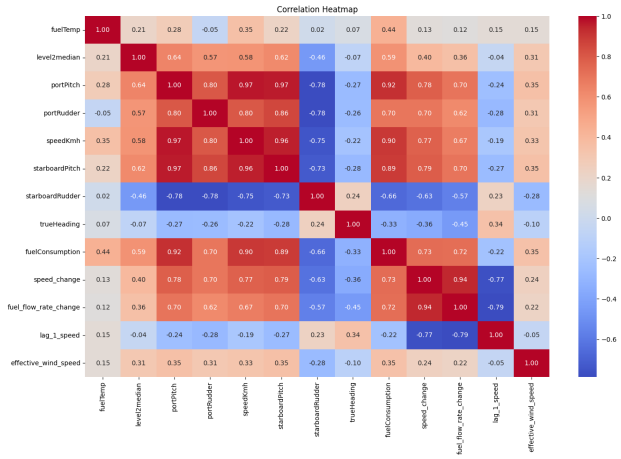

Fig. 2: *Correlation heatmap*

The last part of the data processing was to check for null values and to fix them. In the table III, count of null values for each column are given. They are imputed with the mean of each column, respectively.

Feature	Number of Null Values
fuelTemp	24
level2median	3
portPitch	2
portRudder	2
speedKmh	0
starboardPitch	2
starboardRudder	2
trueHeading	0
fuelConsumption	24
speed_change	1
fuel_flow_rate_change	26
lag_1_speed	1
effective_wind_speed	0

TABLE III: *Number of null values in each feature*

Histograms of all features are given in the figure 3. It can be observed that the majority of features have a skewed distribution.

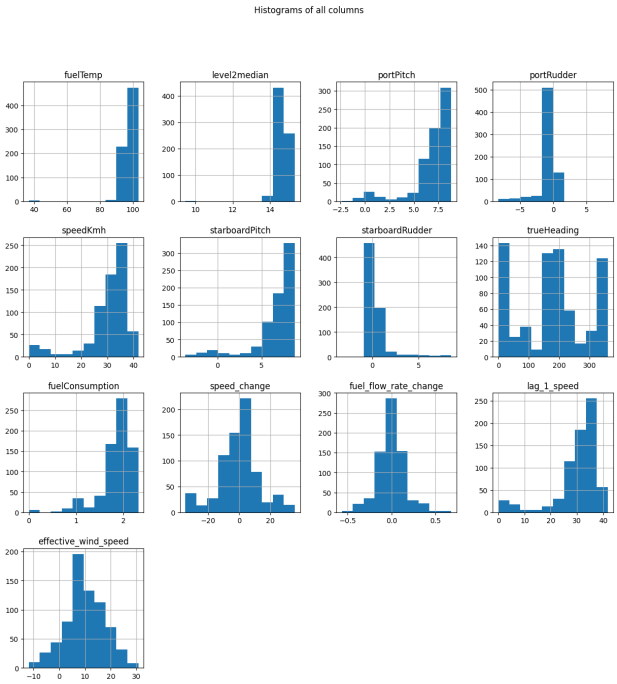

Fig. 3: *Histograms of all features*

Box plots of all features are given in the figure 4. Box plots are usually used to check for outliers, but this dataset requires domain knowledge to be able to say with certainty whether some value is an outlier or a valid case. Since author's domain knowledge was very limited, no outliers were removed.

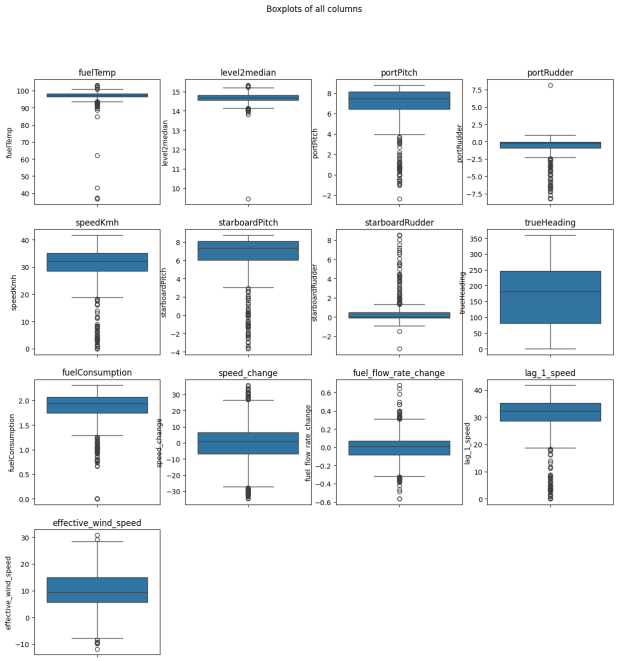

Fig. 4: *Boxplots of all features*

D. Modeling

Three models were chosen for this project: Linear regressor, XGBoost [1], and Random Forest [2]. Linear regressor is a standard model to try first, and XGBoost and Random Forest are known to perform well.

The data was split into training and testing groups, with 80% being used for training, and 20% being used for testing. It can be further discussed if this is enough training data for train-test-split, rather than using cross-validation, but for this project it was decided to stick with a more straight forward approach, which is train-test-split.

Models were compared based on size n, which is the number of decision trees used (where applicable), MSE, which is the Mean Squared Error metric, R^2, which is coefficient of determination, and MAE, which is the Mean Absolute Error of the model.

R^2 (Eq. 1) metric measures how well the regression model fits the data. It represents the proportion of the variance in the dependent variable that is predictable from the independent variable(s).

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{1}$$

- y_i is the actual value of the target variable.
- $\hat{y}_i$ is the predicted value of the target variable.
- $\bar{y}$ is the mean of the actual values.
- n is the number of observations.

The numerator represents the sum of squared residuals (prediction errors), and the denominator represents the total sum of squares (variance of the actual values). An R^2 value closer to 1 indicates a better fit.

MSE (Eq. 2) measures the average squared difference between the actual and predicted values. It indicates how well the regression model performs in terms of prediction accuracy.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2}$$

- y_i is the actual value of the target variable.
- $\hat{y}_i$ is the predicted value of the target variable.
- n is the number of observations.

Lower MSE values indicate better predictive performance, as they imply that the predicted values are closer to the actual values.

MAE (Eq. 3) measures the average absolute difference between the actual and predicted values. It provides an intuitive measure of prediction error, as it represents the average magnitude of errors without considering their direction.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3}$$

- y_i is the actual value of the target variable.
- $\hat{y}_i$ is the predicted value of the target variable.
- n is the number of observations.

Lower MAE values indicate better predictive performance, as they imply that the predicted values are, on average, closer to the actual values.

Number of decision trees used was a hyper-parameter with possible values [100, 200, 300, 400, 500].

III. Data Analysis

Full results overview sorted by R2 is given in the table IV.

Model	R^2 Score	MSE	MAE	Estimators
XGBoost	0.962030	0.073360	0.056210	100
XGBoost	0.962030	0.073360	0.056210	200
XGBoost	0.962030	0.073360	0.056210	300
XGBoost	0.962030	0.073360	0.056210	400
XGBoost	0.962030	0.073360	0.056210	500
Random Forest	0.942952	0.089920	0.062918	400
Random Forest	0.942589	0.090206	0.063051	500
Random Forest	0.942544	0.090241	0.063409	200
Random Forest	0.942038	0.090638	0.063195	300
Random Forest	0.935545	0.095580	0.064608	100
Linear Regression	0.871739	0.134829	0.094163	0

TABLE IV: *Performance comparison of different models*

A. Visualization

Given that both XGBoost and Random Forest were trained with the number of estimators in the set [100, 200, 300, 400, 500], multiple observations can be made.

R2 vs the number of estimators comparison for each model is given in the images 5 and 6. As it can be observed, for Random Forest the best number of estimators is 400. For XGBoost the best number of estimators appears to be 100, but the changes are very small (scale can be seen on the top left corner).

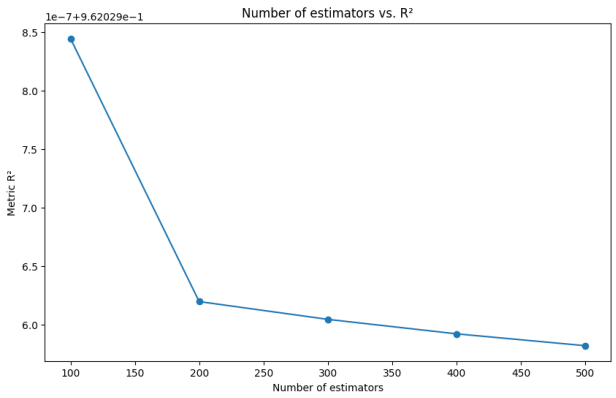

Fig. 5: *R2 by number of estimators for XGBoost Regression*

The best performing model from each of the categories was chosen and the plot representing its predictions versus the actual value is given in the images 7, 8, and 9. Data points have a red gradient, which represents their distance from the actual values (absolute error). The darker the color is, the bigger the error is.

Lastly, the three most important features for the best performing model - XGBoost 100 - are given in the table 10.

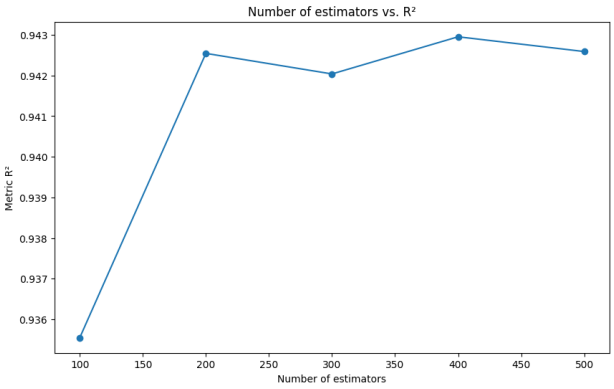

Fig. 6: *R2 by number of estimators for Random Forest Regression*

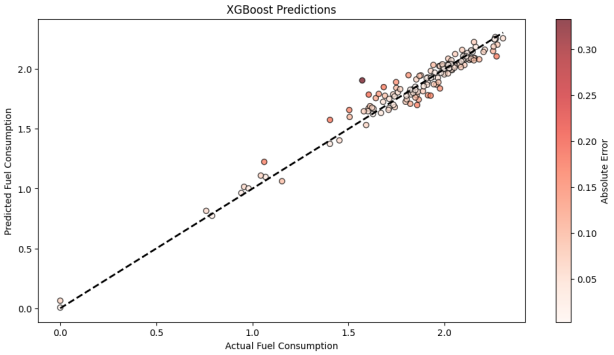

Fig. 7: *Prediction plot for XGBoost Regression*

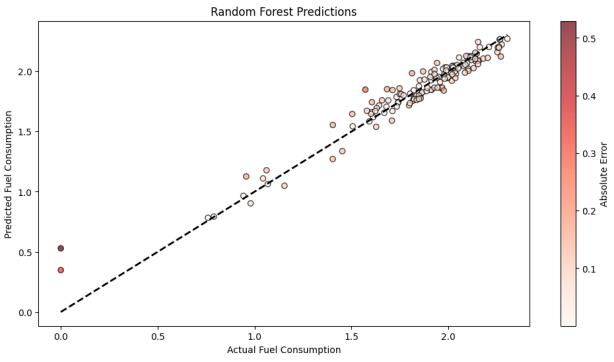

Fig. 8: *Prediction plot for Random Forest*

IV. Conclusion

As it can be seen from the table IV, the R2 score ranges from 0.87 to 0.962, which means all of the models satisfy the requirement of this assignment. It can also be observed that XGBoost outperforms the other two models regardless of the number of estimators used, which simply means this model is better suited for the given data. Below are given some of the possible reasons as to why that is.

- **Boosting vs. Bagging**: XGBoost uses boosting, where trees are built sequentially, and each subsequent tree aims to correct the errors made by the previous ones. This makes it more effective at minimizing prediction errors. Random Forest, on the other hand, uses bagging, where trees are built independently, making it less adaptive to errors.
- **Handling of Over-fitting**: XGBoost has built-in regularization parameters (like lambda and alpha) that help control over-fitting. This can be especially useful in datasets with smaller sizes (like the one used here) or high noise. Random Forest tends to rely on averaging multiple trees to reduce over-fitting.
- **Feature Importance Handling**: XGBoost assigns more importance to features that contribute more to reducing errors. This allows it to better capture the relationships between the most influential features and the target variable.
- **Gradient Descent Optimization**: XGBoost uses gradient descent in its learning algorithm, allowing it to find optimal parameter values for minimizing the loss function more effectively.

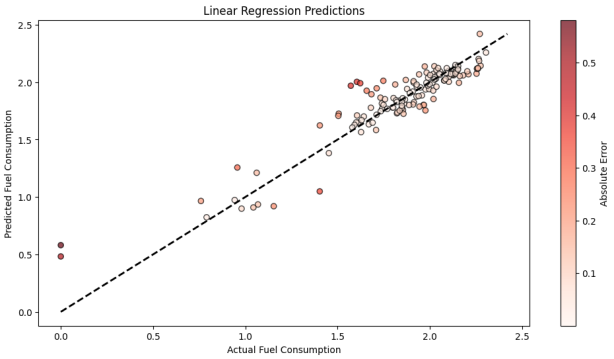

Fig. 9: *Prediction plot for Linear Regressor*

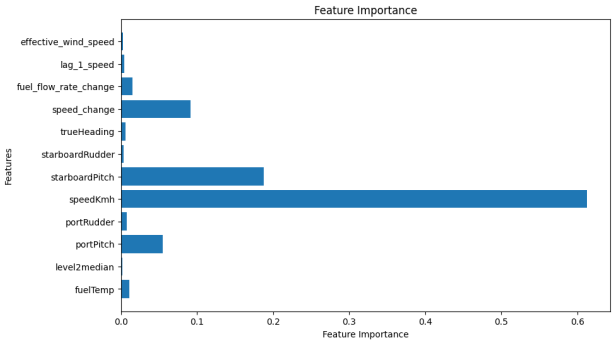

Fig. 10: *Feature importance for the best performing model*

The required accuracy was achieved in the first round of training, and that is most likely due to good chosen features, and the creating of new ones.

It can also be seen from model performance visualizations that models really struggled with two data points for 0 fuel consumption, predicting that as something more. This goes to show that maybe there were not enough examples for the models to learn when fuel consumption should be 0, or the

features didn't indicate that strongly enough.

In conclusion, the greatest challenge here was the domain knowledge. In order to be able to tell whether some data point was an outlier or not, it was required to understand each of the features present in the data. Additionally, to understand whether the correlation of a feature to fuel consumption is a consequence of something logical or just a pure coincidence, also requires domain knowledge. The results here were obtained without any domain knowledge, so they are open to discussions.

REFERENCES

[1] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume 11 of *KDD '16*, page 785–794. ACM, August 2016.

[2] L Breiman. Random forests, October 2001.