Report: Vessel Fuel Consumption Baseline Model and Event Classification System

1. Introduction

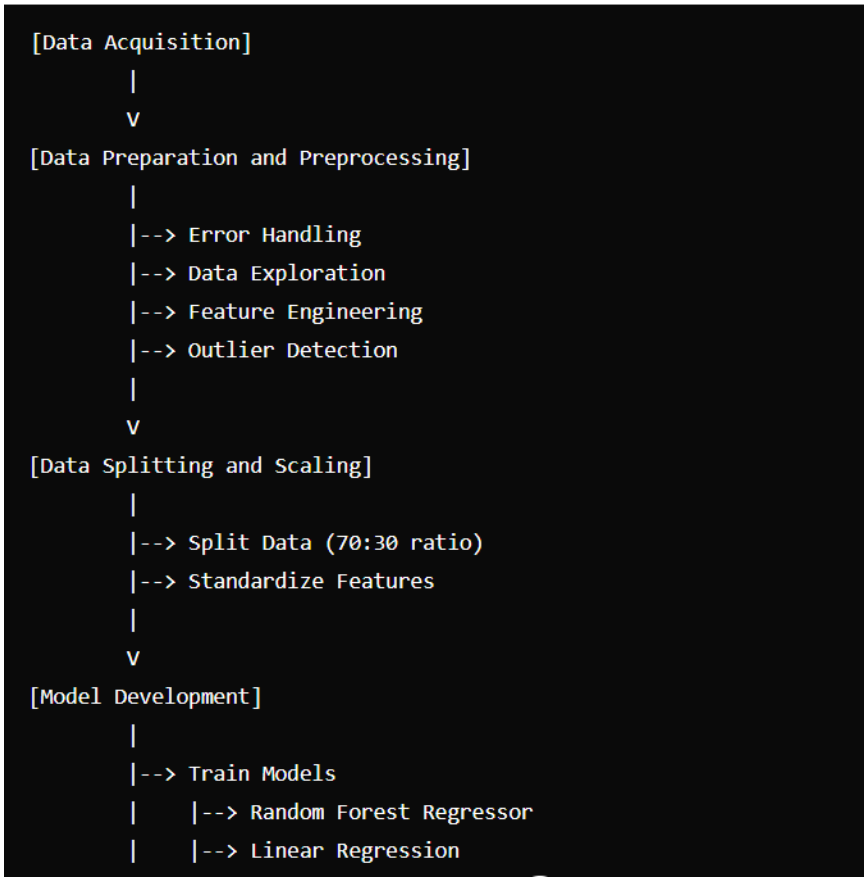
This report outlines the development of a machine learning model to establish a baseline for vessel fuel consumption and a system to classify future consumption events. The project leverages the dataset `vessel_data_2years.csv` to build a predictive model and implement an event classification system based on vessel fuel consumption patterns.

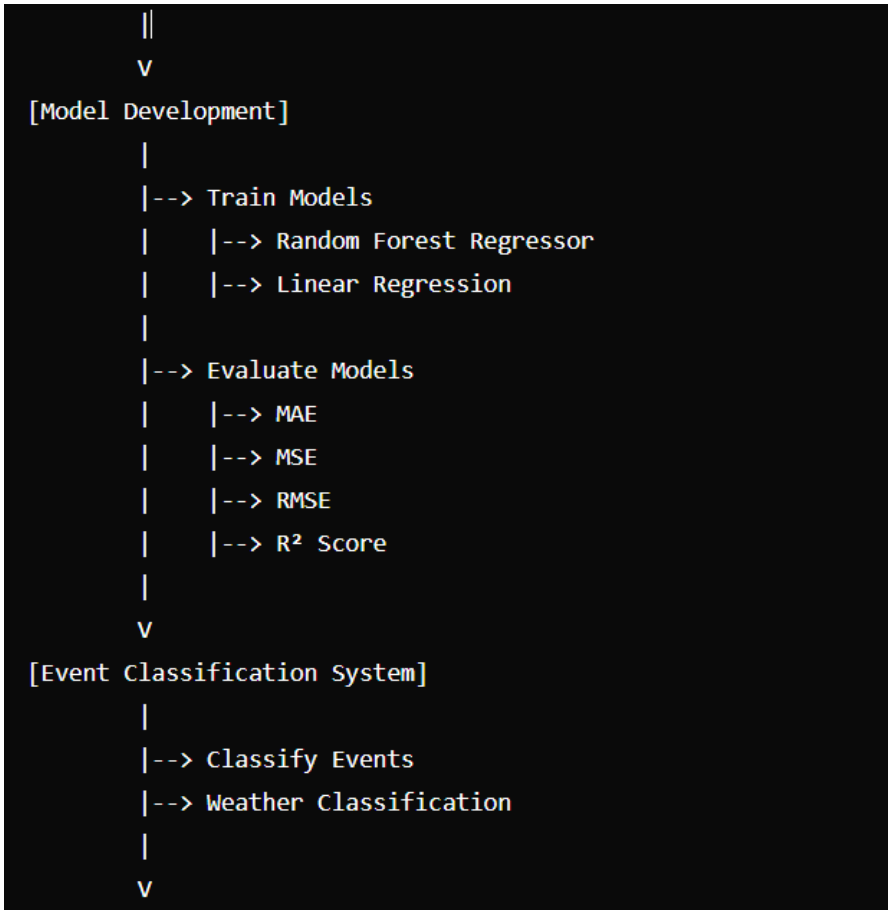
2. Objective

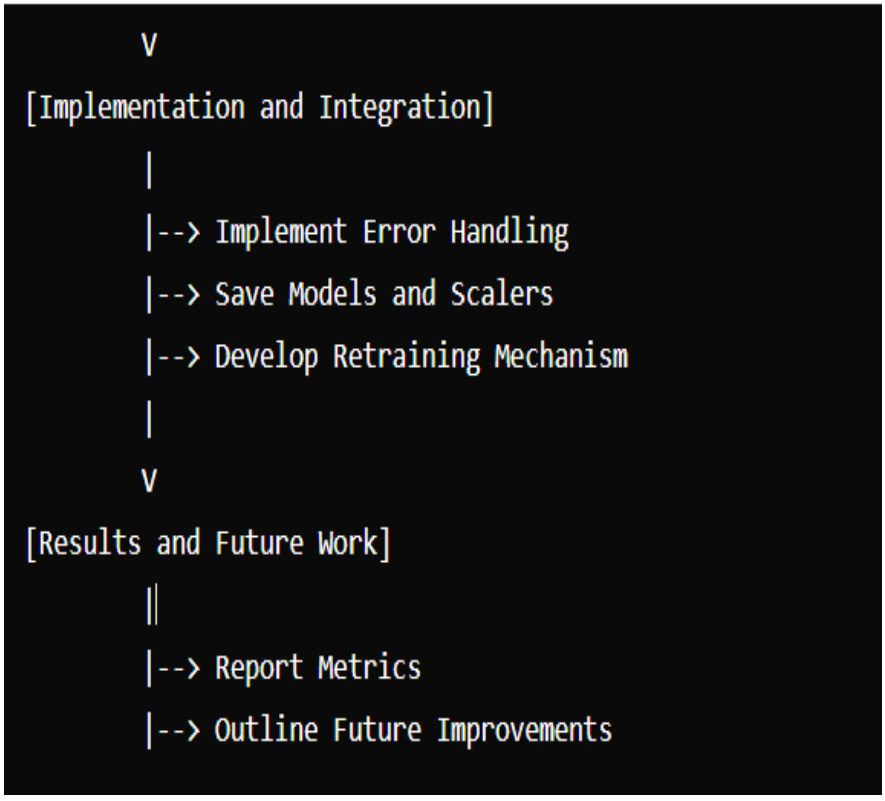
The primary objective of this project is to:

- Develop a robust machine learning model to establish a baseline for vessel fuel consumption.
- Create a system to classify future consumption events based on this baseline.

3. Pipelines




[Model Development]
|--> Train Models
|--> Random Forest Regressor
|--> Linear Regression
|--> Evaluate Models
|--> MAE
|--> MSE
|--> RMSE
|--> R² Score
[Event Classification System]
|--> Classify Events
|--> Weather Classification


[Implementation and Integration]
|--> Implement Error Handling
|--> Save Models and Scalers
|--> Develop Retraining Mechanism
[Results and Future Work]
|--> Report Metrics
|--> Outline Future Improvements

4. Methodology

4.1 Data Preparation and Preprocessing

4.1.1 Data Loading

The dataset `vessel_data_2years.csv` was loaded using pandas with **error handling** to manage potential issues like file not found, empty data, or parsing errors.

```python
def load_data(file_path):
    """Load dataset and handle potential errors."""
    try:
        data = pd.read_csv(file_path)
        return data
    except FileNotFoundError:
        print("Error: File not found.")
        raise
    except pd.errors.EmptyDataError:
        print("Error: No data in file.")
        raise
    except pd.errors.ParserError:
        print("Error: Error parsing the file.")
        raise
```

4.1.2 Data Exploration

Exploratory Data Analysis (EDA) was performed to understand the dataset's structure and content:

- **Initial Data Inspection:**

  Number of rows: 730

  Number of columns: 10

  Columns: `['Date', 'Speed', 'MainEngine_Consumption', 'AuxEngine_Consumption', 'Cargo_Load', 'Sea_State', 'Wind_Speed', 'Season', 'Seasonal_Factor', 'Total_Consumption']`

  Summary statistics and structure were inspected using `data.head()`, `data.info()`, and `data.describe()`.

- **Missing Values:** Checked using `data.isnull().sum()`
- **Duplicates:** Checked using `data.duplicated()`
- **Correlation Analysis:** Visualized using a heatmap.

4.1.3 Feature Engineering

- **Weather Classification:** Classified sea state and wind speed into categorical variables (`Sea_State_Class` and `Wind_Speed_Class`) using custom thresholds.
- **Label Encoding:** Converted categorical weather classifications to numeric values.
- **Feature Selection:** Dropped irrelevant columns based on correlation analysis.
- **Initial Data Inspection**: Used `data.head()`, `data.info()`, and `data.describe()` to understand the dataset's structure and content.
- **Missing Values**: Checked with `data.isnull().sum()` and visualized using a heatmap to identify missing values.
- **Duplicates**: Detected using `data.duplicated().sum()` and removed duplicates with `data.drop_duplicates()`.
- **Correlation Analysis**: Computed correlations with `data.corr()` and visualized using a heatmap.
- **Data Distribution**: Plotted histograms and box plots to understand the distribution and detect outliers in numerical features.
- **Categorical Data Analysis**: Examined frequency counts with `data['categorical_column'].value_counts()` and visualized using bar plots.
- **Feature Relationships**: Explored relationships between features with scatter plots and `sns.pairplot()`.
- **Missing Value Imputation**: Assessed imputation strategies for handling missing values.
- **Data Transformation**: Evaluated the need for scaling or normalizing features.

4.1.4 Outlier Detection and Treatment

Used boxplots and the Interquartile Range (IQR) method to detect and handle outliers. Removed outliers and recalculated quantiles.

```
Number of rows before removing outliers: 730
Number of rows after removing outliers: 698
```

This output shows that there were 730 rows initially, and after removing outliers, the number of rows decreased to 698.

4.2 Model Development

4.2.1 Data Splitting and Scaling

- Split the data into training and testing sets using `train_test_split`.
- Standardized features using `StandardScaler`.
- Used a 70:30 ratio, with 70% of the data allocated for training and 30% for testing

4.2.2 Model Training and Evaluation

- **Random Forest Regressor:** Trained the model and evaluated performance using Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R^2 Score.
- **Linear Regression:** Compared performance with Linear Regression model.

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression

rf = RandomForestRegressor(random_state=101)
rf.fit(x_train, y_train)
```

4.3 Event Classification System

4.3.1 Classification Function

Developed a function to classify events based on predicted consumption and mean consumption. Classified events as "Normal," "Excess," or "Under."

```
def classify_event(features, model, scaler, mean_consumption, threshold=10):
    features_scaled = scaler.transform([features])
    predicted_consumption = model.predict(features_scaled)[0]
    difference = predicted_consumption - mean_consumption
    return "Excess" if difference > threshold else "Under" if difference < -threshold else
```

4.3.2 Weather Classification

Developed a function to classify weather conditions based on sea state and wind speed. Classified sea state into categories: 'Calm,' 'Moderate,' and 'Rough,' and wind speed into categories: 'Light,' 'Moderate,' and 'Strong.'

```
def classify_weather(sea_state, wind_speed):
    # Sea State Classification
    if sea_state <= 2:  # 25th percentile
        sea_state_class = 'calm'
    elif sea_state <= 4:  # 50th percentile
        sea_state_class = 'moderate'
    else:
        sea_state_class = 'rough'

    # Wind Speed Classification
    if wind_speed <= 6.54:  # 25th percentile
        wind_speed_class = 'light'
    elif wind_speed <= 14.32:  # 50th percentile
        wind_speed_class = 'moderate'
    else:
        wind_speed_class = 'strong'

    return sea_state_class, wind_speed_class
```

4.3.3 Model Testing

Provided functions to test both Random Forest and Linear Regression models with new data.

```
def test_models(input_features):
    model, scaler = load_model_and_scaler()
    input_features = np.array(input_features).reshape(1, -1)
    input_features_scaled = scaler.transform(input_features)
    rf_model = model
    rf_prediction = rf_model.predict(input_features_scaled)[0]
    return rf_prediction
```

4.4 Implementation and Integration

- Implemented error handling and validation mechanisms.
- Saved the trained models and scalers for future use.
- Developed a periodic retraining mechanism to update the model with new data.
- **Error Handling in Predictions:** Added error handling to manage issues that may occur during predictions, such as invalid input data or model-related errors.

```
def retrain_model(new_data):
    combined_data = pd.concat([cleaned_data, new_data], ignore_index=True)
    new_features = combined_data.drop(columns=['Total_Consumption'])
    new_target = combined_data['Total_Consumption']
    new_features_scaled = scaler.fit_transform(new_features)
    x_train_new, x_test_new, y_train_new, y_test_new = train_test_split(new_features_scale
    rf_new = RandomForestRegressor(random_state=101)
    rf_new.fit(x_train_new, y_train_new)
    joblib.dump(rf_new, 'consumption_model.joblib')
    joblib.dump(scaler, 'scaler.joblib')
```

5. Results

- **Random Forest Regressor Performance:**
  - Mean Absolute Error (MAE): 2.8655
  - Mean Squared Error (MSE): 12.9575
  - Root Mean Squared Error (RMSE): 3.5997
  - R-squared Score (R^2 Score): 0.8932
- **Linear Regression Performance:**
  - Mean Absolute Error (MAE): 1.3792
  - Mean Squared Error (MSE): 2.9437
  - Root Mean Squared Error (RMSE): 1.7157
  - R-squared Score (R^2 Score): 0.9757

6. Conclusion

The developed vessel fuel consumption baseline model using Random Forest Regressor and Linear Regression models successfully predicts fuel consumption and classifies events. The system is robust and scalable, with the capability for periodic retraining to adapt to new data.

7. Future Work

- **Model Optimization:** Explore hyperparameter tuning to enhance model performance.
- **Real-time Integration:** Implement real-time data processing and classification for dynamic scenarios.
- **Advanced Analytics:** Incorporate additional features or advanced algorithms to improve predictions.

8. References

- Vessel Fuel Consumption Dataset: `vessel_data_2years.csv`
- Scikit-learn Documentation: https://scikit-learn.org/
- Pandas Documentation: https://pandas.pydata.org/
- Matplotlib Documentation: https://matplotlib.org/
- Seaborn Documentation: https://seaborn.pydata.org/