Documentation

Notebooks:
Multiple notebooks were used to preprocess the data, train the models, evaluate the results, and compute the data for the Decision Support System. For each notebook a short description will be given. The notebooks themselves also contain further explanations and descriptions of what was done.

- Preprocessing: In this notebook the original three datasets are loaded and preprocessed according to the description in the final report. The output of this notebook is four dataframes: (1) the historic dataset, (2) the dataset used for modeling, (3) the preprocessed ports dataset, and (4) the preprocessed vessel dataset. The last two datasets (3) and (4) contain newly added features that were computed based on the historic dataset.
- Port Prediction: In the Port Prediction notebook the preprocessed datasets are used to train the LightGBM Ranker. However, some more preprocessing of the dataset was needed due to the nature of the model. Furthermore, new features were added. After feature selection and hyperparameter tuning the model was trained and evaluated on the test set. The predictions are saved to be used in the Results notebook. Lastly a baseline model is created for which the predictions are also saved.
- Travel Duration: In this notebook the same steps of adding features, feature selection, hyperparameter tuning and training the model are taken. Except this time the model is a LightGBM Regressor. The predictions of the model and the baseline are saved.
- Stay Duration: Exactly the same was done for predicting the stay duration. This notebook is almost exactly the same as the Travel Duration notebook, except for the fact that the stay duration was predicted instead of the travel duration. The predictions of both the model and the baseline were saved.
- Results: The Results notebook loads the predictions for all three objectives and computes the evaluation metrics and plots that were used in the report.
- DSS: In this notebook the predictions are made for the DSS. Only for the visits where there is no target information (i.e. the last visit of each vessel) were used. The data was processed the same way as was done in the Port Prediction/Travel Duration/Stay Duration notebooks. For each objective the model and selected features were loaded and were used to predict the next port, the travel duration and the stay duration. Lastly, only the relevant information was selected and saved to be used in the DSS.

Decision Support System:
The DSS is made with Python, a Python module for dashboarding named Dash and a .css stylesheet. These three modules together form an interactive application with a few general layout components and a few callback functions that change the output of these layout components. The .css (cascading style sheet) file contains classes which define the styling of the components in the dashboard. A description of the Python file that produces the application is given below.

Filename: *main.py*
Content:

- Initalise data: two data files named ‘port_info.csv’ and ‘DSS_input.csv’ are read and stored in Pandas DataFrame objects.
- Default parameters worldmap: the global variable ‘fig’ is assigned as default figure for the worldmap, which will be a key element of the application.
- Creating app: the app is assigned as Dash application
- App layout: the layout of the application is being created with all components that will be present on the dashboard. The app layout consists of four components:
    - Banner component
    - Tab selection component
    - Dropdown component
    - Output component
        - Worldmap
        - Output tables
        - Output figure
- App callbacks: callback functions make it possible to have interactivity between the user, the data and the output components. Each callback consists of at least one Output variable, at least one Input variable and optionally a ‘State’ variable that gives the state of a layout component as an extra input variable. These are the callback functions:
    - *Update_output:*

```
@app.callback(
    Output('output_table','children'),
    Output('ports_figure','children'),
    Input('dropdown','value'),
    State('tab_selection','value')
)
def update_output(dropdown_value,selected_tab):
    ''' This function updates the information of the output table and
output figure when there occurs a change in the dropdown element. '''
```

- *Reset_dropdown:*

```
@app.callback(
    Output('label','children'),
    Output('dropdown','options'),
    Output('dropdown','placeholder'),
    Output('dropdown','value'),
    Input('tab_selection','value')
)
def reset_dropdown(selected_tab):
    ''' This function resets the dropdown options and clears dropdown
selection as soon as the user switches the tab. '''
```

- *Update_worldmap:*

```
@app.callback(
    Output('worldmap','figure'),
    Input('dropdown','value'),
    State('tab_selection', 'value')
)
def update_worldmap(dropdown_value,selected_tab):
    ''' This function updates the worldmap graph when there occurs a
change in the dropdown element.'''
```