Leren & Beslissen

CargoProbe

Predicting the arrival of shipping containers

Author:
David Werkhoven
13909495
Jacomijn Prins
13418874
Paul Koole
11765763
Sem Kjaer
12374040

Supervisor:
Robin Puthli

Teaching Assistent:
Mitchell Verhaar

Universiteit van Amsterdam

CARGOPROBE

February 5, 2023

1 Abstract

For the last few years the global supply chain has been dealing with the delays caused by the fallout of the Covid-19 pandemic. These delays cause many disruptions throughout the entire supply chain. Team four from the University of Amsterdam was tasked with predicting the accuracy of the Estimated Time of Arrival (ETA) given by CargoProbe using Machine Learning techniques. The data set provided by CargoProbe was thoroughly analysed to properly clean and understand the data set. Multiple algorithms were trained, but the random forest outperformed the other regression models. With a Mean Squared Error of 7.92 days and a Mean Absolute Error of 1.2 days, the predictions given by the algorithm greatly improved the ETA provided by CargoProbe.

The results will help CargoProbe improve their prediction of the Actual Time of Arrival (ATA) for their client, reducing the negative impacts of delays in the supply chain. Additionally giving a starting point for further research into improving these predictions even further.

Contents

2 Introduction

2.1 Problem Definition

Gathering data about container delivery times is crucial for supply chains to function properly. This is especially true in a time where global supply chains are still dealing with the fallout of the covid-19 disruption as well as a slew of other problems like rising energy costs [3].

CargoProbe is a Dutch company that specializes in tracking supply chains. They follow containers being shipped all over the world for their clients and make a comprehensive overview of their status. This information can then be accessed by their clients in a direct and uniform way. They provide their clients with real time updates about how far along the container is, what activities it is currently going through and most importantly an estimation of when the container will arrive [2].

As a part of this, they gather data directly from shipping companies to get the most accurate and relevant data using their large network of shipping companies. The estimated time of a container's arrival is currently based on heuristics given by the shipping company.

They provided team four with a part of their data set on their Amazon web server. The data set shows all the values that they provide their clients with.

2.2 Challenge

The main problem Cargo Probe currently has is that their estimated time of arrival of a shipping container in their data set is not accurate enough for them. They have asked team four to analyze how accurate their current the Estimated Time of Arrival (ETA) really is, and if they could perhaps increase the accuracy of this value by using machine learning. An accurate ETA is very important to CargoProbe due to the hindering role of unexpected delays in their client's supply chains. Knowing better when a container will arrive could save their clients from extra costs, or even allow them to take earlier action to solve upcoming supply problems. The challenges that team four will thus have to tackle are:

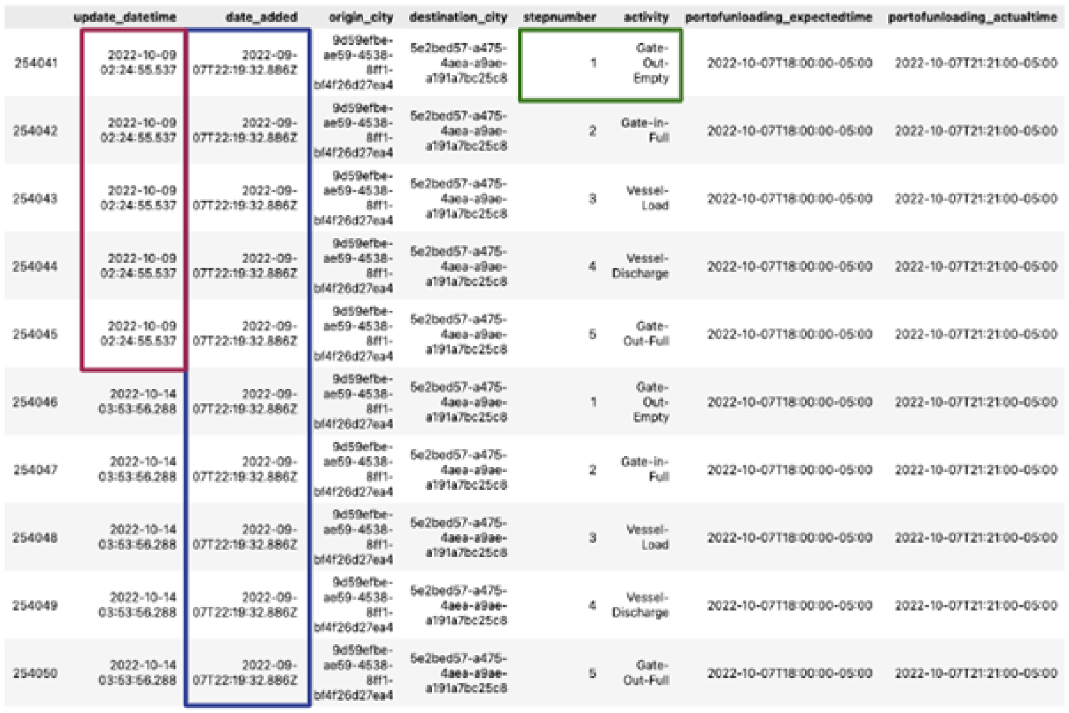
1. Calculating the accuracy of the ETA provided by CargoProbe
2. Cleaning the data set and adding features to fit a model
3. Train the model to accurately predict the delay of the ETA provided by CargoProbe

3 Dataset

3.1 Original Data

The data was stored on a server that team four had remote access to. The original data test consists of:
331956 rows
23 features

Structure of the data:
Every row of the data set is an entry for a specific container at a specific activity. During the trip a container will most likely go through multiple activities, numbered by step number. During the trip there may be multiple updates which will repeat the previous step numbers. Figure 1 shows a small incomplete fraction of the data set, where you can see the structure. This figure shows one trip in which the container got from its start position to its destination, shown by the `date_added` column having one value (blue). During this trip two updates were given, shown by the two sections in the `update_date` column (red). During the update the data shows five steps with different activities that the container went through (green).



	update_datetime	date_added	origin_city	destination_city	stepnumber	activity	portofunloading_expectedtime	portofunloading_actualtime
254041	2022-10-09 02:24:55.537	2022-09-07T22:19:32.886Z	9d59efbe-ae59-4538-8ff1-bf4f26d27ea4	5e2bed57-a475-4aea-a9ae-a191a7bc25c8	1	Gate-Out-Empty	2022-10-07T18:00:00-05:00	2022-10-07T21:21:00-05:00
254042	2022-10-09 02:24:55.537	2022-09-07T22:19:32.886Z	9d59efbe-ae59-4538-8ff1-bf4f26d27ea4	5e2bed57-a475-4aea-a9ae-a191a7bc25c8	2	Gate-in-Full	2022-10-07T18:00:00-05:00	2022-10-07T21:21:00-05:00
254043	2022-10-09 02:24:55.537	2022-09-07T22:19:32.886Z	9d59efbe-ae59-4538-8ff1-bf4f26d27ea4	5e2bed57-a475-4aea-a9ae-a191a7bc25c8	3	Vessel-Load	2022-10-07T18:00:00-05:00	2022-10-07T21:21:00-05:00
254044	2022-10-09 02:24:55.537	2022-09-07T22:19:32.886Z	9d59efbe-ae59-4538-8ff1-bf4f26d27ea4	5e2bed57-a475-4aea-a9ae-a191a7bc25c8	4	Vessel-Discharge	2022-10-07T18:00:00-05:00	2022-10-07T21:21:00-05:00
254045	2022-10-09 02:24:55.537	2022-09-07T22:19:32.886Z	9d59efbe-ae59-4538-8ff1-bf4f26d27ea4	5e2bed57-a475-4aea-a9ae-a191a7bc25c8	5	Gate-Out-Full	2022-10-07T18:00:00-05:00	2022-10-07T21:21:00-05:00
254046	2022-10-14 03:53:56.288	2022-09-07T22:19:32.886Z	9d59efbe-ae59-4538-8ff1-bf4f26d27ea4	5e2bed57-a475-4aea-a9ae-a191a7bc25c8	1	Gate-Out-Empty	2022-10-07T18:00:00-05:00	2022-10-07T21:21:00-05:00
254047	2022-10-14 03:53:56.288	2022-09-07T22:19:32.886Z	9d59efbe-ae59-4538-8ff1-bf4f26d27ea4	5e2bed57-a475-4aea-a9ae-a191a7bc25c8	2	Gate-in-Full	2022-10-07T18:00:00-05:00	2022-10-07T21:21:00-05:00
254048	2022-10-14 03:53:56.288	2022-09-07T22:19:32.886Z	9d59efbe-ae59-4538-8ff1-bf4f26d27ea4	5e2bed57-a475-4aea-a9ae-a191a7bc25c8	3	Vessel-Load	2022-10-07T18:00:00-05:00	2022-10-07T21:21:00-05:00
254049	2022-10-14 03:53:56.288	2022-09-07T22:19:32.886Z	9d59efbe-ae59-4538-8ff1-bf4f26d27ea4	5e2bed57-a475-4aea-a9ae-a191a7bc25c8	4	Vessel-Discharge	2022-10-07T18:00:00-05:00	2022-10-07T21:21:00-05:00
254050	2022-10-14 03:53:56.288	2022-09-07T22:19:32.886Z	9d59efbe-ae59-4538-8ff1-bf4f26d27ea4	5e2bed57-a475-4aea-a9ae-a191a7bc25c8	5	Gate-Out-Full	2022-10-07T18:00:00-05:00	2022-10-07T21:21:00-05:00

Figure 1: Section from original data set

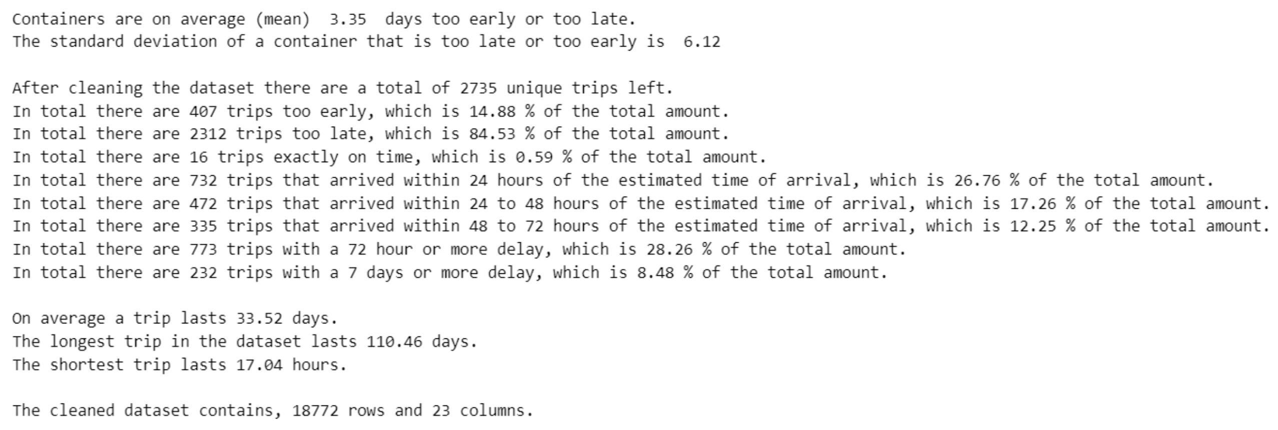
Missing values:
The way this data was gathered has a big impact on the lack of values in the data set. Cargoprobe explained that many of the entries are manually put into the data set. The ETA's are provided by the shipping companies during the transportation of the container. These are the reasons why almost half of the data consists of NaN values. The section data cleaning will show how the NaN values were cleaned from the data set.

3.2 Data exploration

This section shows certain statistics of the data set. These statistics were measured after the cleaning (which will be explained in the next section) was completed. The exploration was done before, but for this report it is good to be able to compare the difference in the delay with the data that team four actually trained on.

3.2.1 Stats

These are a few statistics of the data set, all these values are based on the values team four used for their model. So it does not contain all the rows the original data set contained.



```
Containers are on average (mean)  3.35  days too early or too late.
The standard deviation of a container that is too late or too early is  6.12

After cleaning the dataset there are a total of 2735 unique trips left.
In total there are 407 trips too early, which is 14.88 % of the total amount.
In total there are 2312 trips too late, which is 84.53 % of the total amount.
In total there are 16 trips exactly on time, which is 0.59 % of the total amount.
In total there are 732 trips that arrived within 24 hours of the estimated time of arrival, which is 26.76 % of the total amount.
In total there are 472 trips that arrived within 24 to 48 hours of the estimated time of arrival, which is 17.26 % of the total amount.
In total there are 335 trips that arrived within 48 to 72 hours of the estimated time of arrival, which is 12.25 % of the total amount.
In total there are 773 trips with a 72 hour or more delay, which is 28.26 % of the total amount.
In total there are 232 trips with a 7 days or more delay, which is 8.48 % of the total amount.

On average a trip lasts 33.52 days.
The longest trip in the dataset lasts 110.46 days.
The shortest trip lasts 17.04 hours.

The cleaned dataset contains, 18772 rows and 23 columns.
```

Figure 2: The statistics of the cleaned data set

Figure 3 shows the distribution of the length of all the container trips. It shows most of the trips had a duration between 20 and 40 days. This provides context for most of the trips arriving within a week of the ETA, as shorter trips = are less likely to exceed a delay of more than a week.

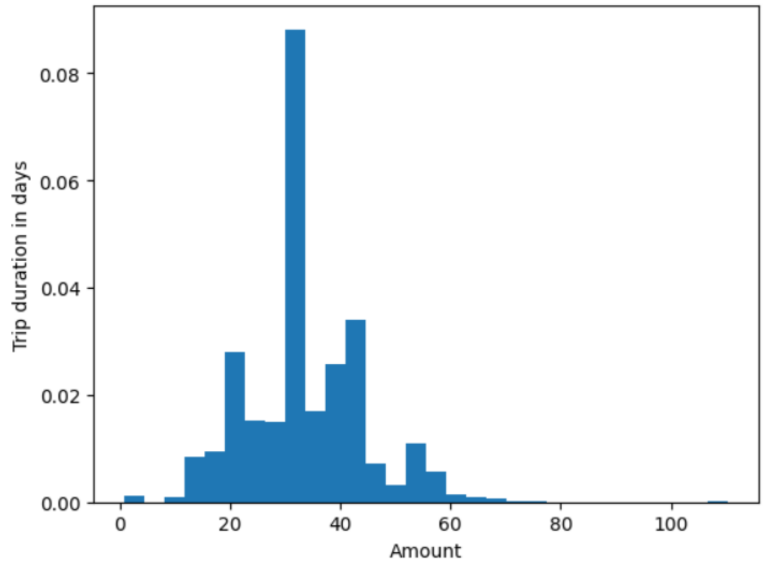


Figure 3: The distribution of the trip duration over the entire cleaned data set.

3.2.2 CargoProbe delay

To be able to predict the delay of a container, the delay from CargoProbe has to be visualized. The delay is the difference between the actual time of arrival (ATA), `portofunloading_actualtime` in the data set, and the expected time of arrival (ETA), `portofunloading_expectedtime` in the data set, of a certain container. A container can be both too early and too late. The supervisor, Robin Puthli, from CargoProbe explained that a delay becomes a problem when it is more than 3 days, as from that point the client will have to pay an extra storing fee. Thus being able to accurately predict when a container will be later than 3 days will be of high importance, to make sure a client has time to arrange for a different pick up time and will not be forced to pay an extra fee.

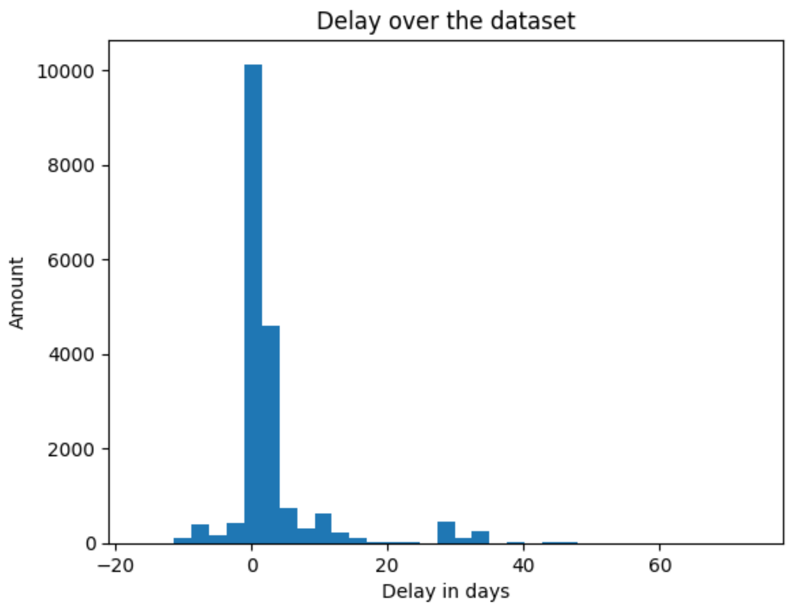


Figure 4: The distribution of the delay over the entire cleaned data set.

When looking at the distribution of the delay from CargoProbe (see figure 4), it seems that CargoProbe is already fairly accurate when it comes to predicting the ATA, as most predictions fall within the 3 day range. However the average delay from CargoProbe (taking the absolute value to take into account the early containers) is 3.58 days. 23.89 % of the containers fall outside of this 3 day margin and have an average delay of 11 days which is when the delay becomes a real problem.

3.3 Data cleaning

Before the missing values were filled, columns with certain data types were converted to more generic data types.

A visual example of the total amount of NaN values in the data set.
See appendix, figure 10

All columns that contained string values got converted to lowercase, this was done to remove duplicate and misspelled values.

The timestamps in the columns that contained timestamps all got converted to UTC time, this way the entire data set used the same definition of time rather than their local time zones.

Step number got converted to integer, because the steps of an update in the data set are always whole numbers.

After those simple changes the missing NaN values were filled. This was mostly done by obtaining the data from other columns and rows. Because the data set is manually filled by port workers it often occurs that values in rows are left unfilled. It however is possible to combine rows with different missing values to create a fully filled row. This technique was used to fill all rows that did not contain timestamps.

The results of these changes were as followed.

See appendix, figure 11

The timestamp columns with missing values were mostly dropped because they were not needed to solve the challenge. All trips that did not arrive were also dropped because the information they contained was not relevant for solving the challenge.

The most important timestamp column that contained missing values was the portofunloading_expectedtime column. This column was an important column because it was needed to create the target time_left column which will be explained in feature engineering. An attempt was made to fill this column with multiple techniques, the results of those techniques are as followed.

See appendix, figure 14

After comparing these results the conclusion was made that filling the ETA values would not be a smart choice with such big offsets from the real ETA values. Due to that, the choice was made to fill the ETA values with ffill which is a function that takes the ETA value of the previous row. This was done per trip, to not take the ETA of a different trip.

To properly use the data set for machine learning the duplicate updates had to be removed, a trip contained a lot of updates which were duplicate rows from the update that happened beforehand. This could cause the machine learning model to weigh an update more than another due to it occurring more than once. To solve this issue only the relevant row of an update was kept. The following line of code was used for this:

```
df.loc[(df['activity_actualtime'].notna())].drop_duplicates(subset=
['update_datetime', 'date_added'], keep='last')
```

A visual of the update changes can be found in figure 12 and figure 13

The end results of filling the data set were as followed, see appendix, figure 15

3.4 Feature engineering

1. `Time_left`
   This is the time that a container has left to go based on the difference between the ETA that they currently have assigned to them and the time of the update in which that ETA is provided. This time value is created using the following formula, eta - update_datetime

2. `Time_spent`
   This is the amount of time that has been spent so far during the trip, this time value

is created using the following formula, update_datetime - date_added.

3. `Vessel_arrived`
   This is a binary value. 1 means the vessel has arrived in the port of unloading and still needs to unload the container. 0 means the vessel has not yet arrived at the port of unloading. This value was calculated by looking if the activity_city was equal to the portofunloading_city

4. `Delay`
   This is simply the difference between the `portofunloading_eta` en `portofunloading_ata`. Given the threshold of 72 hours we will be able to say a container is late or on time. The value of this column is calculated using the following formula, portofunloading_actualtime - portofunloading_expectedtime

3.4.1 One Hot Encoding

To be able to perform regression on the data set that contains many categorical variables, we performed One Hot Encoding (OHE). OHE assigns each unique value for each variable a different column, filled with zero's and one's for each row as to whether that value was assigned to that specific row. Using all the given features would have resulted in an unmanageable large data set, so we decided to only encode a handful of features:

1. `origin_city`
2. `vesselname`
3. `activity`
4. `activity_source`
5. `activity_city`
6. `portofunloading_city`

These features were selected on giving the best results, but with the least chance of over fitting. the feature `vesselname` for instance, gave a very good score, but based on the number of vessels, this would likely be a variable that would lead to over fitting.
OHE the given features and adding in the `delay` and the `time_left` features, resulted in a data frame containing 18630 rows and 384 columns.

4 Methods

The main objective as stated in the challenge is to find a way to improve the ETA that CargoProbe provides to predict the arrival of the containers of their customers. As we base our prediction on the ETA that CargoProbe gets from the shipping.

At the start of the project after the data cleaning we tried out multiple different models to find the one that would suit the data the best and give the best prediction. In this chapter we explain the different models we thought of and tried out and give the reason why we decided to not use them. And we explain why we chose our final model.

4.1 ML preprocessing

The Cargo Probe dataset contained missing ETA values, while cleaning the dataset a lot of values were already lost due to removing every trip that contained no actual time of arrival. And since the ETA was already an estimation the conclusion came to mind that these values could potentially be filled using machine learning or other missing value algorithms.

4.1.1 K Nearest neigbors

K nearest neighbors is a machine learning algorithm which can be used to fill missing values in a dataset. The algorithm matches a point within a multi dimensional space with K of its neighbors to estimate the missing value.[5]
K nearest neighbors is mainly based on two parameters:

- Number of neighbors The number of neighbors is the number of neighbors the model will use to compare with to estimate the missing value. A low amount will create less general results and a high amount causes local effects to not be shown as much. A good starting point for the amount of neighbors is the Square root of the amount of data points.
- Distance function The distance function is used to decide which data points are seen as neighbors, the most straightforward distance function is the euclidean distance, this distance function measures the distance between two points in a straight line. Other options of distance functions are Cosine similarity, correlation, Minowsky and Chi square.

During the filling of the time_left column whilst using K nearest neighbors a k value of 130 was used. The distance metric used was the euclidean distance.

4.2 Considered methods - Predicting CargoProbe delay

To improve the ETA given by CargoProbe we want to calculate the error of the ETA that they give compared to the actual ATA. This error we call the delay which is calculated in days. We have trained multiple regression models to try and find the one best suitable for the challenge. For the regression models we try to predict the delay that the ETA given by CargoProbe would produce.

4.2.1 Linear Regression

Linear regression is a model that looks at the linear correlation between independent variables and the dependent one [4].
To use multivariate linear regression one has to make a few assumptions:

1. Normality: To be able to perform a linear regression the data has to be normally distributed. As this plot shows the distribution of the delay between the ETA and the ATA is, although not perfectly, normally distributed.

2. Independence of independent variables: To check for the independence between the independent variables, we made a correlation matrix between all the columns and dropped the ones that had a correlation > 0.95.

3. Linearity: Most likely the correlations between the independent and dependent variables are not fully linear, so this model will most likely not be the best one to describe the correlations and give the best predictions.

4.2.2 Polynomial regression

Polynomial regression uses the same assumptions as linear regression except for the linear correlation, as a model of polynomial regression can model for curvilinear correlations. This would be a valuable difference to the linear regression as mentioned, the correlation will most likely be more complex than a simple linear one.
However polynomial regression is highly dependent on the order you choose. Thus fitting the model would take a lot of training and making sure the model was not overfitting. Due to the size of our dataset after the one hot encoding it would have taken a lot of computer power to compute even a simple second polynomial and due to the quality of our own hardware, we were not able to. Even with dropping certain variables to minimize the number of columns after the one hot encoding this problem was still in effect.

4.3 Final method - Predicting CargoProbe delay

4.3.1 Random forest

A random forest is an ensemble machine learning algorithm that generates several different decision trees. A decision tree is a machine learning algorithm that creates a tree that leads to a number of 'terminal' of 'leaf' nodes which determine output that the input will be assigned. At every node in the tree, it 'splits' the training data, based on the variable that will create the highest information gan (IG). This is iteratively applied until a point is reached where all the training data had the same target variable, or until a pre-defined maximum depth is reached. Each decision tree is constructed using a random subset of the features as well as a random sample of the training data, thus finding many different connections between the independent variables. The random forest takes all these different decision trees and will compute an output, after which it will take the average of all the outputs to produce the final output of the forest. Random forests were introduced by Leo Breiman introduced by Leo Breiman in his 2001 paper, 'Random Forests'. [1]

Random Forest is considered a robust algorithm with fast training times and the ability to handle large datasets. It also performs very well compared to other algoritms, being often involved in winning solutions for machine learning contests. This algorithm will take the complexity of the correlations between the variables and consequently produce a much better fitting algorithm to the data, however this might also result in over fitting. In order to combat this we dropped certain features that the algorithm was too depended on and dropped columns that had too much correlation.

Random forests can use categorical as well as numerical data to split on. So from a purely mechanical point of view that is naive to certain practical problems in the data (e.g. values that appear only once, values that are hard not to overfit on when determining a train and test set), all features that we have could be used to split on.

However, we did not calculate feature importances because of our one-hot encoding method. This method splits up all categorical values in lots of new ones. By doing so, the overview is lost.

Another advantage is the interpretability of the algorithm. Breiman introduced a way to calculate the importance of every feature used, by calculating the information gain (IG) of every split and aggregating that information over all trees used.[1] CargoProbe can use this information especially in any further research if they want to know or be able to explain why certain containers are expected to have more delay.

4.4 Implemantation

For the implementations of the algorithms the python machine learning library SciKit-learn (referred to as sklearn) was used.

4.4.1 Training-Test Split

Sklearn has a function, `train_test_split`, that splits the data set into a training and test set (consisting of X values, the independent variables, and y values, the dependent or target values) given a certain distribution. Before it splits into these train en test data sets it will randomize the rows. This is done to combat the discrepancy in the distribution between the training and test data set of the variable, on which the data set is ordered.
However in this case splitting the data based on the index variable without shuffling is a favorable outcome. The data is indexed on the `date_added` column, which means the trips are in order. Shuffling this order before splitting, will result in the updates belonging to the same trip occurring in both the training and test set. This can lead to the algorithm recognizing a certain container and correctly calculating the delay without finding the actual underlying correlations. Thus the training-test split was done without shuffling, with a 80-20 distribution.

As was made visible in section 3.2.2 the distribution of delays is highly unequal. To reduce this inequality and prevent data imbalance that will influence the outcome, the peak on delays falling within the 0 to 3 day range was reduced by 20%. This was to ensure enough data to train on, but to minimize the effect of dataimblance on the outcome of the algorithm.

4.4.2 Randomized search

To find the best parameters for the Random forest model the sklearn function, `RandomizedSearchCV` was used. The randomized search function is given an range in parameters and will train using all the different combinations of parameters and will produce the optimal ones. The final results of the random forest were based on the following parameters:

- number of estimators: 150
- min samples to split: 4
- min samples per leaf: 3
- max features: 'auto'
- max depth: 136
- bootstrap: True

This resulted in a slight increase of the Mean Absolute Error, but a decrease in the Mean Square Error (explained further in the next section). This is optimal as it shows that the model becomes better for the outliers in its prediction.

4.5 Validation

1. Mean Squared Error:
   The Mean Squared Error (MSE) calculates the mean of the squared error between the value of the test set and the value predicted by the model, using the formula:

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

   with $\tilde{y}_i$ being the target value and y_i being the prediction.

   Because of the squaring of the error, MSE will prescribe more weight to bigger errors and thus be more perceptive to outliers in the error.

2. Mean Absolute Error: The Mean Absolute Error (MAE) calculates the mean of the absolute error, using the formula:

   $$MAE = \frac{1}{n}\sum_{i=1}^{n}|\tilde{y}_i - y_i|$$

   with $\tilde{y}_i$ being the target value and y_i being the prediction.

   Taking the absolute of the error will take into account the fact that a container can also be too early.

3. Max error: The Max Error shows the largest difference between the true value and the prediction. This will help to show whether an output contains large outliers.

5 Results - Predicting CargoProbe delay

5.1 CargoProbe delay

To be able to show the accuracy of our models in predicting the delay from CargoProbe, figure 5 shows the distribution of the delay from CargoProbe in the testset:

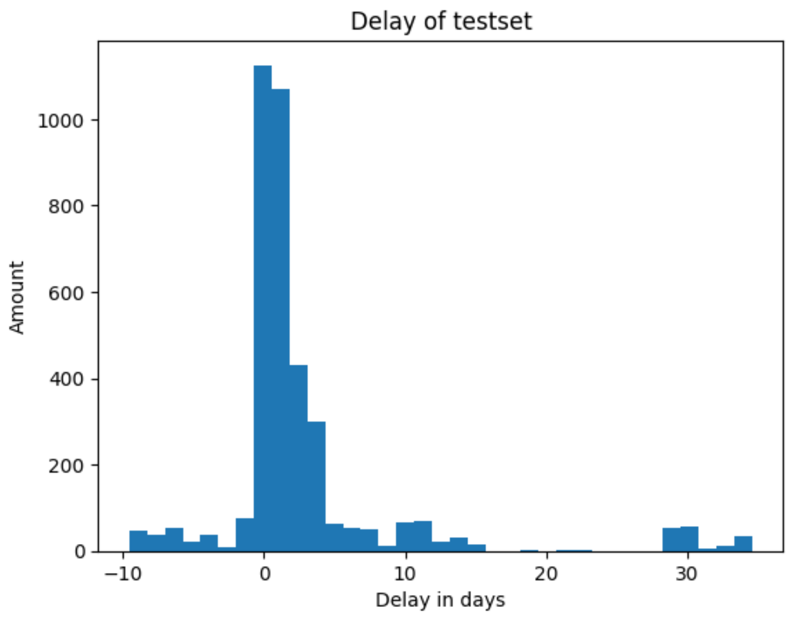


Figure 5: The distribution of the delay over the testset.

5.2 Linear Regression

The Linear regression did not provide the best model. In the early stages of training it was already apparent that the model was too simple to show the correct correlations. The metrics were very high:

- [MAE:] $1.87 \cdot e^7$
- [MSE:] $5.24 \cdot e^{17}$
- [MAX:] $3.11 \cdot e^9$

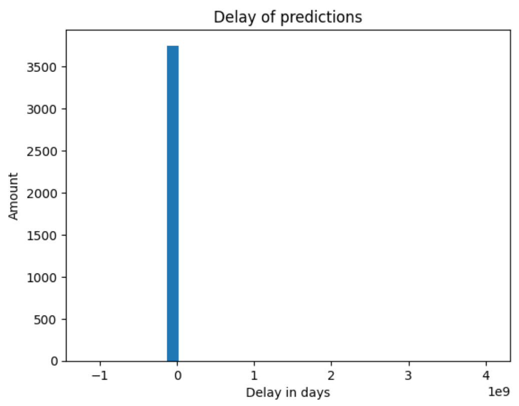


Figure 6: The distribution of the delay of prediction made by the linear regression model.

The explanation for this would be that the model was severely under fitting the data. As seen in figure 6 the model only gives one prediction.

5.3 Random forest

After training and changing the parameters and OHE features the final validation measures are listed below.

- [MAE:] 1.20
- [MSE:] 7.92
- [MAX:] 29.82

To better grasp the performance of the model we will compare it's results to the ETA provided with the data enhanced as discussed in the pre-processing section. The same metrics for the enhanced ETA over the test sets conclude as follows.

- [MAE:] 3.58
- [MSE:] 54.75
- [MAX:] 34.58

These metrics show that on average the random forest model is of by 1.20 days when predicting the delay of a given container, a definite improvement over the MAE of 3.58 for the original ETA. The relatively low MSE shows that the predictions do not contain many big outliers at least compared to the original prediction. The maximum error is smaller as well.

Given that data can be biased we can get accurate results even when not making sensible predictions. Predicting the mean value for every entry might result in decent accuracy scores but that does not make for a very nuanced model. The predicted delays can be seen in the next figure.

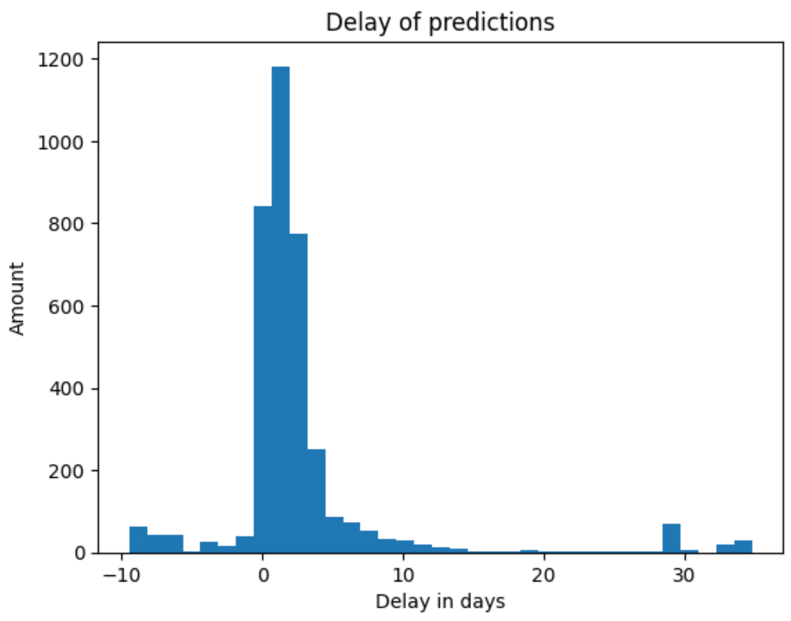


Figure 7: The distribution of predictions of the delay made by the random forest model.

The distribution appear to closely resemble the original distribution of the enhanced delay. Hence we conclude that the model is not only making accurate predictions but is also reasonably unbiased.

The distribution of the original delay shows that there are small clusters of outliers. Looking at the distribution of the model result error we can see whether the model suffers the same fate.

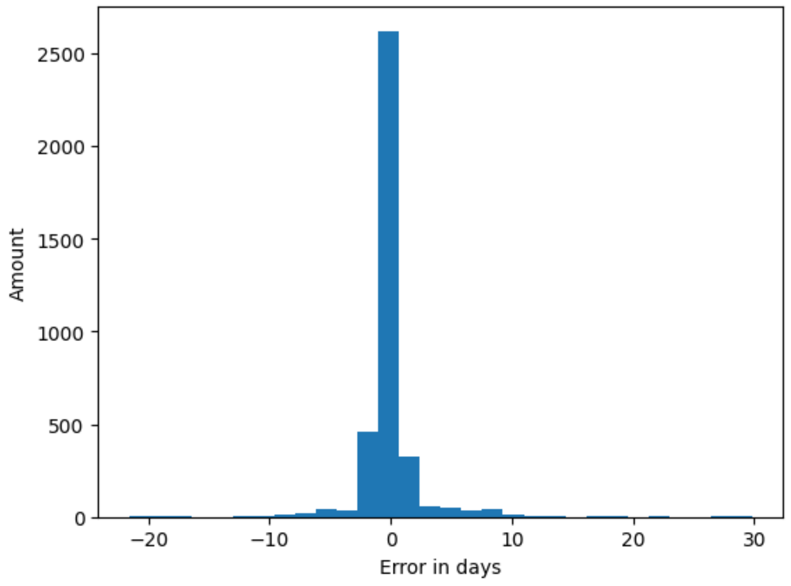


Figure 8: Model error in days.

The model does not show a sizeable cluster of errors around the 30 day mark. It shows an improvement upon many of the original predictions that were highly inaccurate. There are still some outliers in the results however they seem less prevalent as can also be construed from the significantly improved MSE.

The accuracy of our model is bound to be influenced by the time left until a container is expected to arrive. It seems only natural that predictions over months will be less accurate compared to predictions over days. Figure 9 compares the MSE of our predictions with the enhanced ETA by weeks until a container is expected to arrive. This will show us how the duration over which a prediction is made influences the accuracy of our model and how this relation compares to the original ETA.

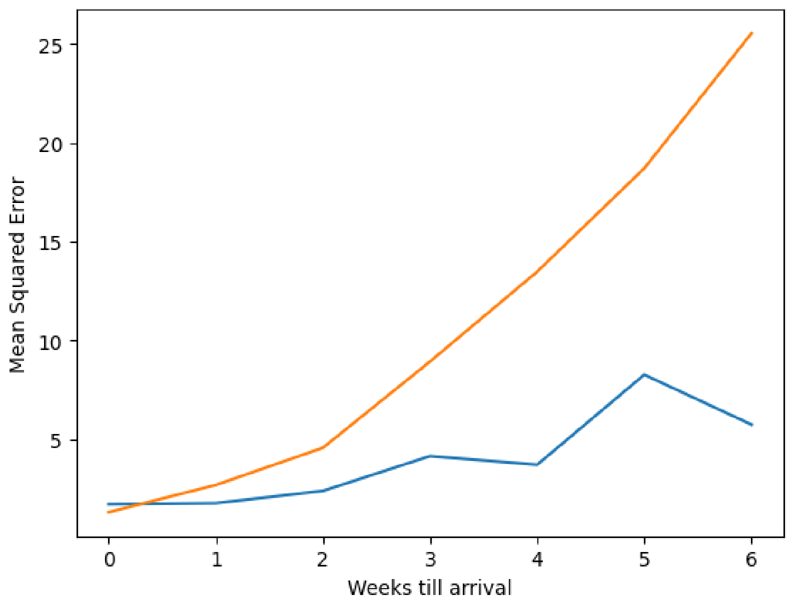


Figure 9: Mean Squared Error of the model based on time till arrival.

Evidently, the error of our results do increases with the duration over which predictions are made. However the same is true for the original ETA. Moreover figure 9 shows us that that this increase is much more profound for the original ETA compared to the model's prediction. Thus our model performs significantly better when making predictions for containers that are far away from the destination compared to the provided ETA. The ETA however, slightly overtakes our model in terms of accuracy within one week from delivery. However as mentioned, these first few days have an error margin as the first 3 days do not pose any problem for CargoProbe.

6 Conclusion

Team 4 has implemented several techniques within data manipulation and machine learning in order to tackle the challenge provided by CargoProbe. To improve the prediction that they can give their clients about the arrival time of their container.

To tackle the first two challenges, the data set provided by CargoProbe was meticulously analysed to understand its structure and to clean it to provide a proper data set to train the algorithms without losing to many data points. Besides calculating the accuracy of the predictions from CargoProbe itself in order to create a framework on which could be improved.

To provide a prediction of the delay of the given ETA by CargoProbe team 4 considered different methods. Linear and polynomial regression seemed like possible options, but soon linear regression showed to produce highly under fitted outputs. Polynomial regression was thus attempted, but was too computationally taxing for the data and hardware available for the project.

The final algorithm chosen was the random forest, for its ability to capture the different correlations between the given features. The random forest is very precise, but measures had to be taken to make sure that the features are not over fitting to the data. The variable `vesselname` was dropped, because of dangers of over fitting. And the data was split in order to prevent updates from the same trip to influence the accuracy in the test set.

7 Discussion

The results of the model trained in this project are already of good use to CargoProbe as the accuracy of the model falls within the span of 3 days which they identified as the range in which they would have no problems. However as the project consisted only of 4 weeks and due to the amount of cleaning needed for the data set, with more time this algorithm could be improved upon. Besides this, there are a few thing CargoProbe could improve upon in further research:

Feature importance
Due to the One Hot Encoding (OHE) of the data set it is difficult to perform a precise feature importance analysis. As every unique value in the columns that are OHE is given its own column and thus becomes a feature itself. Features such as `vesselname`, were identified as features that were causing some over fitting. Still, to be able to undeniably say that all the features used, have no influence on over fitting, further analysis should be conducted. Feature analysis over OHE columns is not impossible, but was unfeasible for the 4 week scope of this project.

Training
The chronological train-test split was used to simulate the way in which the algorithm could be used in practice; training on the last $\tilde{3}$ months to then predict the next week of containers arriving. The random forest was trained on data over a longer period of time and used to predict containers over a longer period of time, but with more data the algorithm could be trained on data over a shorter period of time to perhaps take into account more of the current correlations at that time.

Reason for outliers
At last CargoProbe could look into analysing the model for reasons why the outliers occur. Due to the scope of the project and the difficulty of analysing a random forest to see on which feature splits it predicts outliers, this rapport does not include this analysis. However this could be very useful for CargoProbe in order to even prevent long delays instead of only

predict them in advance.

Different algorithm
This project looked into the random forest regression algorithm, however there are many other algorithms that could suit this problem and might provide another good way of solving it.

ML preprocessing
K-Nearest Neighbours was used to provide a way in which CargoProbe could use the prediction of the accuracy of the delay even if no delay was provided through the data by producing the ETA through the KNN algorithm. The algorithm could not produce usable output within the scope of this project. Improving the algorithm could be done in further research by using other distance metrics than the euclidean distance or by attempting to train the model on different K values.

8 Appendix

References

[1] Leo Breiman. “Random forests”. In: *Machine learning* 45 (2001), pp. 5–32.

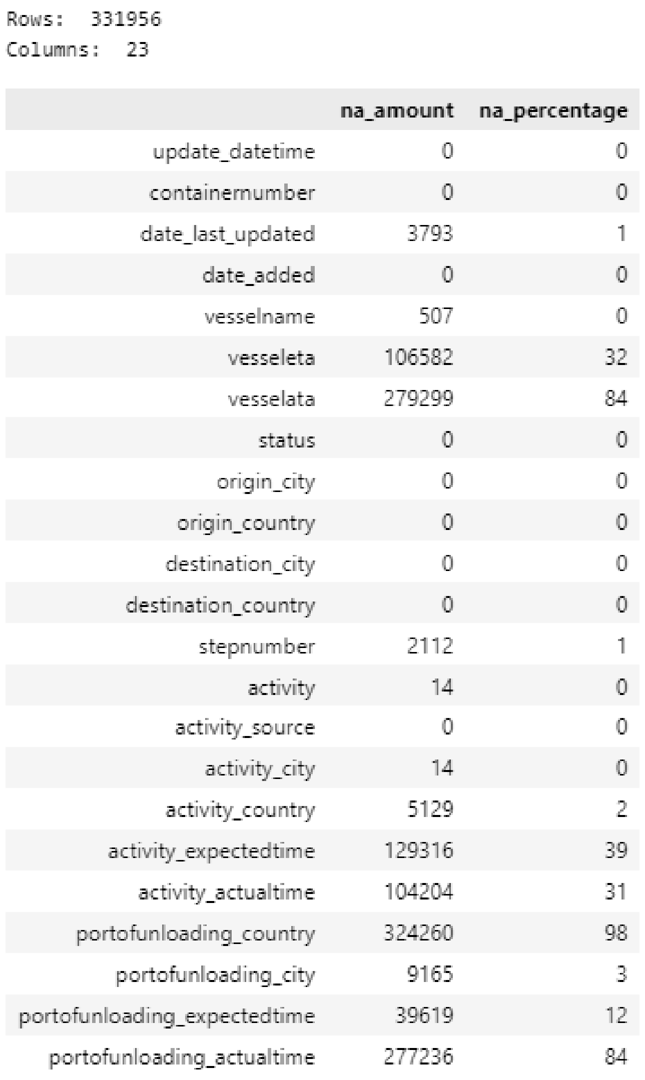
[2] *CargoProbe — Digital Supply Chain Visibility*. Accessed on: 2023-02-02. 2023. URL: `https://www.cargoprobe.com/`.

[3] *Global Logistics 2023: Supply chains under pressure*. `https://www.logisticsmgmt.com/article/global_logistics_2023_supply_chains_under_pressure`. Accessed: 2023-2-5.

[4] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning*. Springer Series in Statistics. New York, NY: Springer, Mar. 2009.

[5] Leif E Peterson. “K-nearest neighbor”. In: *Scholarpedia* 4.2 (), p. 1883.

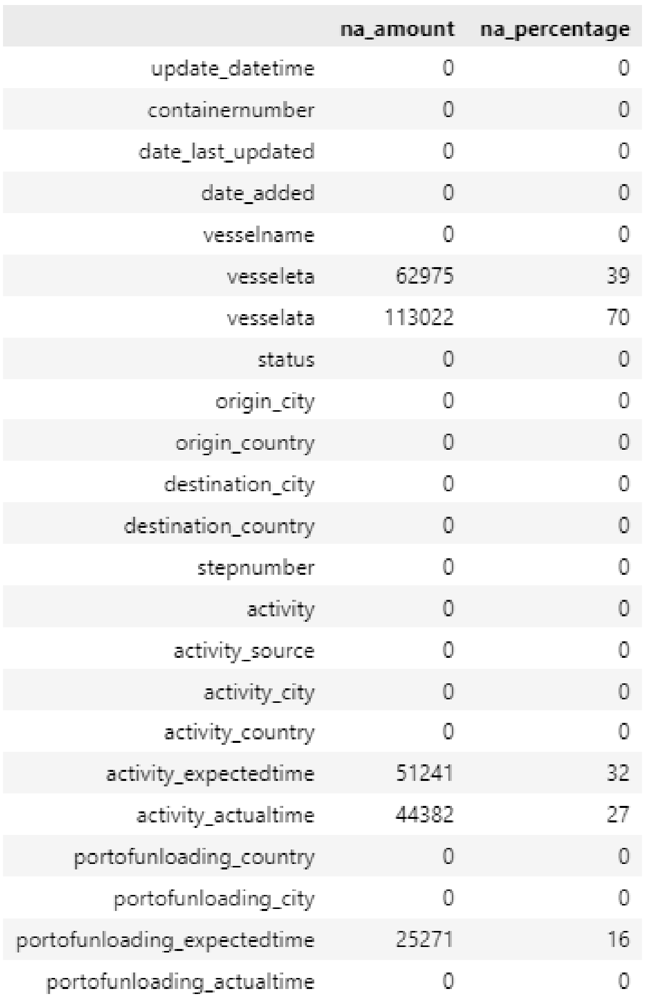
8.1 Figures



```
Rows:  331956
Columns:  23
```

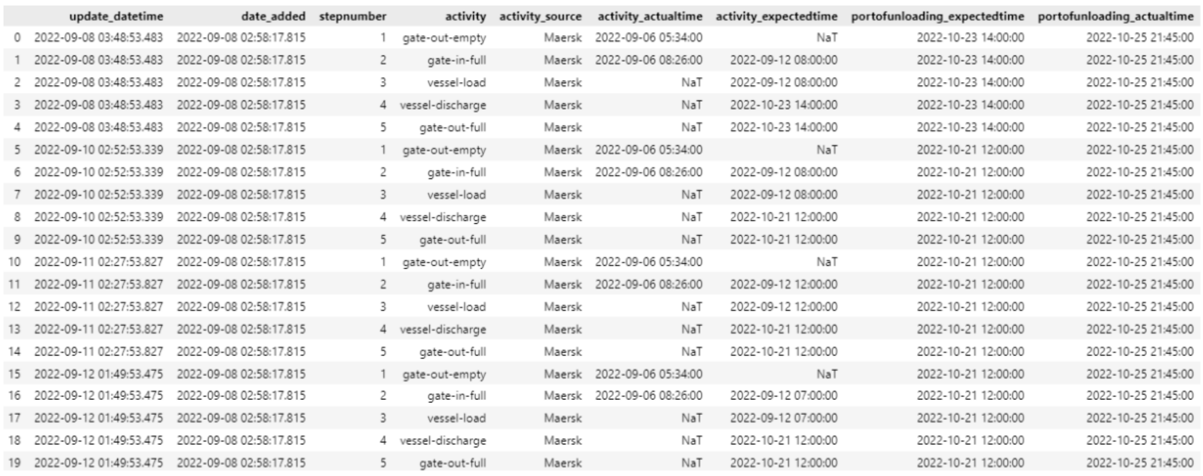
	na_amount	na_percentage
update_datetime	0	0
containernumber	0	0
date_last_updated	3793	1
date_added	0	0
vesselname	507	0
vesseleta	106582	32
vesselata	279299	84
status	0	0
origin_city	0	0
origin_country	0	0
destination_city	0	0
destination_country	0	0
stepnumber	2112	1
activity	14	0
activity_source	0	0
activity_city	14	0
activity_country	5129	2
activity_expectedtime	129316	39
activity_actualtime	104204	31
portofunloading_country	324260	98
portofunloading_city	9165	3
portofunloading_expectedtime	39619	12
portofunloading_actualtime	277236	84

Figure 10: The data set before filling any NaN values.



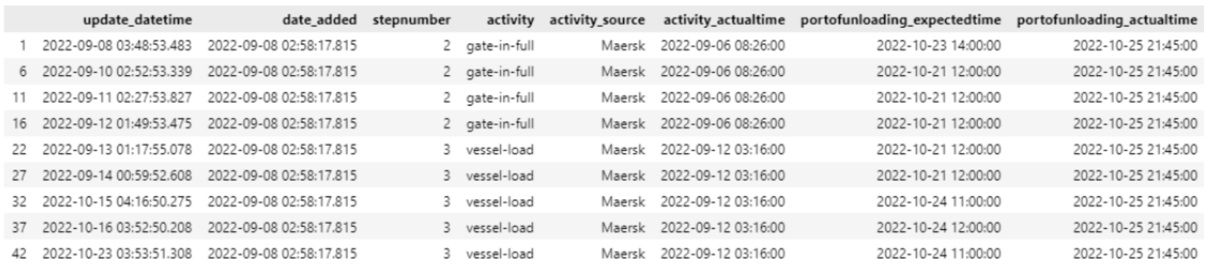
	na_amount	na_percentage
update_datetime	0	0
containernumber	0	0
date_last_updated	0	0
date_added	0	0
vesselname	0	0
vesseleta	62975	39
vesselata	113022	70
status	0	0
origin_city	0	0
origin_country	0	0
destination_city	0	0
destination_country	0	0
stepnumber	0	0
activity	0	0
activity_source	0	0
activity_city	0	0
activity_country	0	0
activity_expectedtime	51241	32
activity_actualtime	44382	27
portofunloading_country	0	0
portofunloading_city	0	0
portofunloading_expectedtime	25271	16
portofunloading_actualtime	0	0

Figure 11: The data set after filling non timestamp NaN values.



	update_datetime	date_added	stepnumber	activity	activity_source	activity_actualtime	activity_expectedtime	portofunloading_expectedtime	portofunloading_actualtime
0	2022-09-08 03:48:53.483	2022-09-08 02:58:17.815	1	gate-out-empty	Maersk	2022-09-06 05:34:00	NaT	2022-10-23 14:00:00	2022-10-25 21:45:00
1	2022-09-08 03:48:53.483	2022-09-08 02:58:17.815	2	gate-in-full	Maersk	2022-09-06 08:26:00	2022-09-12 08:00:00	2022-10-23 14:00:00	2022-10-25 21:45:00
2	2022-09-08 03:48:53.483	2022-09-08 02:58:17.815	3	vessel-load	Maersk	NaT	2022-09-12 08:00:00	2022-10-23 14:00:00	2022-10-25 21:45:00
3	2022-09-08 03:48:53.483	2022-09-08 02:58:17.815	4	vessel-discharge	Maersk	NaT	2022-10-23 14:00:00	2022-10-23 14:00:00	2022-10-25 21:45:00
4	2022-09-08 03:48:53.483	2022-09-08 02:58:17.815	5	gate-out-full	Maersk	NaT	2022-10-23 14:00:00	2022-10-23 14:00:00	2022-10-25 21:45:00
5	2022-09-10 02:52:53.339	2022-09-08 02:58:17.815	1	gate-out-empty	Maersk	2022-09-06 05:34:00	NaT	2022-10-21 12:00:00	2022-10-25 21:45:00
6	2022-09-10 02:52:53.339	2022-09-08 02:58:17.815	2	gate-in-full	Maersk	2022-09-06 08:26:00	2022-09-12 08:00:00	2022-10-21 12:00:00	2022-10-25 21:45:00
7	2022-09-10 02:52:53.339	2022-09-08 02:58:17.815	3	vessel-load	Maersk	NaT	2022-09-12 08:00:00	2022-10-21 12:00:00	2022-10-25 21:45:00
8	2022-09-10 02:52:53.339	2022-09-08 02:58:17.815	4	vessel-discharge	Maersk	NaT	2022-10-21 12:00:00	2022-10-21 12:00:00	2022-10-25 21:45:00
9	2022-09-10 02:52:53.339	2022-09-08 02:58:17.815	5	gate-out-full	Maersk	NaT	2022-10-21 12:00:00	2022-10-21 12:00:00	2022-10-25 21:45:00
10	2022-09-11 02:27:53.827	2022-09-08 02:58:17.815	1	gate-out-empty	Maersk	2022-09-06 05:34:00	NaT	2022-10-21 12:00:00	2022-10-25 21:45:00
11	2022-09-11 02:27:53.827	2022-09-08 02:58:17.815	2	gate-in-full	Maersk	2022-09-06 08:26:00	2022-09-12 12:00:00	2022-10-21 12:00:00	2022-10-25 21:45:00
12	2022-09-11 02:27:53.827	2022-09-08 02:58:17.815	3	vessel-load	Maersk	NaT	2022-09-12 12:00:00	2022-10-21 12:00:00	2022-10-25 21:45:00
13	2022-09-11 02:27:53.827	2022-09-08 02:58:17.815	4	vessel-discharge	Maersk	NaT	2022-10-21 12:00:00	2022-10-21 12:00:00	2022-10-25 21:45:00
14	2022-09-11 02:27:53.827	2022-09-08 02:58:17.815	5	gate-out-full	Maersk	NaT	2022-10-21 12:00:00	2022-10-21 12:00:00	2022-10-25 21:45:00
15	2022-09-12 01:49:53.475	2022-09-08 02:58:17.815	1	gate-out-empty	Maersk	2022-09-06 05:34:00	NaT	2022-10-21 12:00:00	2022-10-25 21:45:00
16	2022-09-12 01:49:53.475	2022-09-08 02:58:17.815	2	gate-in-full	Maersk	2022-09-06 08:26:00	2022-09-12 07:00:00	2022-10-21 12:00:00	2022-10-25 21:45:00
17	2022-09-12 01:49:53.475	2022-09-08 02:58:17.815	3	vessel-load	Maersk	NaT	2022-09-12 07:00:00	2022-10-21 12:00:00	2022-10-25 21:45:00
18	2022-09-12 01:49:53.475	2022-09-08 02:58:17.815	4	vessel-discharge	Maersk	NaT	2022-10-21 12:00:00	2022-10-21 12:00:00	2022-10-25 21:45:00
19	2022-09-12 01:49:53.475	2022-09-08 02:58:17.815	5	gate-out-full	Maersk	NaT	2022-10-21 12:00:00	2022-10-21 12:00:00	2022-10-25 21:45:00

Figure 12: Update before merging.



	update_datetime	date_added	stepnumber	activity	activity_source	activity_actualtime	portofunloading_expectedtime	portofunloading_actualtime
1	2022-09-08 03:48:53.483	2022-09-08 02:58:17.815	2	gate-in-full	Maersk	2022-09-06 08:26:00	2022-10-23 14:00:00	2022-10-25 21:45:00
6	2022-09-10 02:52:53.339	2022-09-08 02:58:17.815	2	gate-in-full	Maersk	2022-09-06 08:26:00	2022-10-21 12:00:00	2022-10-25 21:45:00
11	2022-09-11 02:27:53.827	2022-09-08 02:58:17.815	2	gate-in-full	Maersk	2022-09-06 08:26:00	2022-10-21 12:00:00	2022-10-25 21:45:00
16	2022-09-12 01:49:53.475	2022-09-08 02:58:17.815	2	gate-in-full	Maersk	2022-09-06 08:26:00	2022-10-21 12:00:00	2022-10-25 21:45:00
22	2022-09-13 01:17:55.078	2022-09-08 02:58:17.815	3	vessel-load	Maersk	2022-09-12 03:16:00	2022-10-21 12:00:00	2022-10-25 21:45:00
27	2022-09-14 00:59:52.608	2022-09-08 02:58:17.815	3	vessel-load	Maersk	2022-09-12 03:16:00	2022-10-21 12:00:00	2022-10-25 21:45:00
32	2022-10-15 04:16:50.275	2022-09-08 02:58:17.815	3	vessel-load	Maersk	2022-09-12 03:16:00	2022-10-24 11:00:00	2022-10-25 21:45:00
37	2022-10-16 03:52:50.208	2022-09-08 02:58:17.815	3	vessel-load	Maersk	2022-09-12 03:16:00	2022-10-24 12:00:00	2022-10-25 21:45:00
42	2022-10-23 03:53:51.308	2022-09-08 02:58:17.815	3	vessel-load	Maersk	2022-09-12 03:16:00	2022-10-24 11:00:00	2022-10-25 21:45:00

Figure 13: Update after merging.

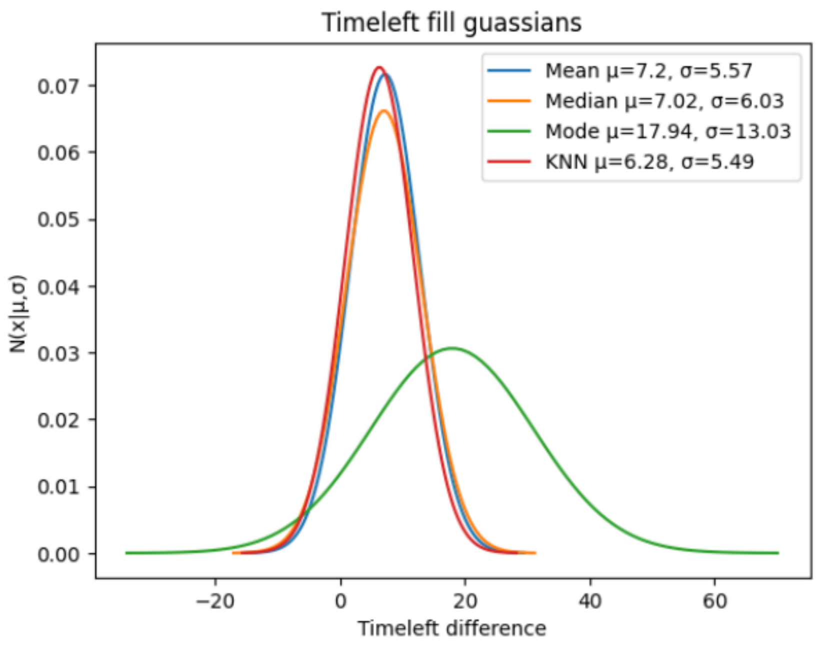
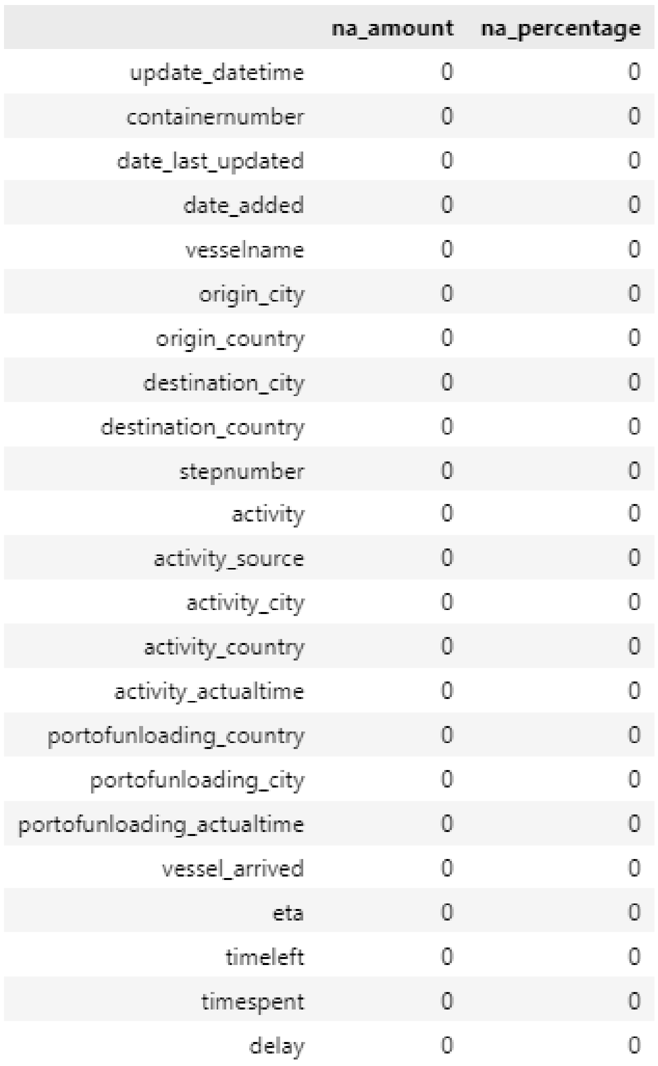


Figure 14: The comparison of results from filling the timeleft column.



	na_amount	na_percentage
update_datetime	0	0
containernumber	0	0
date_last_updated	0	0
date_added	0	0
vesselname	0	0
origin_city	0	0
origin_country	0	0
destination_city	0	0
destination_country	0	0
stepnumber	0	0
activity	0	0
activity_source	0	0
activity_city	0	0
activity_country	0	0
activity_actualtime	0	0
portofunloading_country	0	0
portofunloading_city	0	0
portofunloading_actualtime	0	0
vessel_arrived	0	0
eta	0	0
timeleft	0	0
timespent	0	0
delay	0	0

Figure 15: The data set after dropping and filling timestamp NaN values.