ML Internship Assignment: Shipping Technologies

Vamsi Yerramsetti, BSc Computer Science student at Radboud University
Supervisor: David Woudenberg, Technical Lead at Shipping Technologies

May 27, 2022

1 Introduction

I was asked to analyze one week's worth of sensor data from a single inland barge ship of length 136 meters and do the following:

1. Understand and preprocess the data.
2. Build and train the data on a model.
3. Predict the $rot - realised$ (rotational acceleration) in a time-series approach.
4. Evaluate the model
5. Report the findings.

2 Analyzing the data via csv files

I first read through the assignment and tried to understand the meaning behind the columns. I did not know some of the terms so I had to google some of the terms such as draft cog and how thee relate to each other. I then took some time to go through the CSV files. I realized that some files had more data entries than the others like "st-1a2090-2021-26-rot-realized.csv ". However, all CSV files had the date and trip id. I found this to be crucial later when I did the data split. I also realized that the date and trip-id for all the CSV files were synchronized, so this was nice.

The assignment said I had to join the data in the correct way. I realized that the position.csv file has the bulk of the data and has most of the parameters needed to build the model. So I created a new csv file called shippingtechupdated.csv where I joined the draft, rot-realised, and rudder to the position.csv. I then loaded this into the jupyter notebook. Another initial problem I faced later (but I am mentioning it now) is that my computer was hanging when I ran the code on the notebook quite frequently despite update attempts. I took another look at the shippingtech.csv files and realized that the rot-rudder column had way more entries than the position file and thought it would be best if I shorten it so it matches the rest of the position.csv columns (479814). I know this wasn't the best practice but I thought this shortcut would help me deal with dealing with the data.

3 Analyzing and Prepossessing the data via jupyter notebook:

As I mentioned before I loaded the notebook along with some basic panda, NumPy, and plotting imports. I first got some statistical insight into the data I'm working with such as mean, count, min, max, etc. The mean and count were interesting to me as they painted an image of the volume and range of data I'm working with. Next, I got a list of the different data types for each respective column. Most columns have a float data type. I then wanted to find out how many null values were present in the dataset. Missing values or their replacement values can lead to huge errors in your analysis output whether it is a machine learning model. However, while I am writing this report I realized that testing on a K-nearest /Naive Bayes model may have been nice as they are somewhat resistant to null values. Anyway, I found out that the draft column had 309780 null values. I will

explain how I dealt with this further down in the report. I also found the number of unique values in each column just to get an idea of the mode for each data category in the dataset.

I then checked for categorical attributes and tried removing trip-id from it. This didn't benefit me much. Now coming back to the null values in draft, there are many ways to deal with null values in ML models such as dropping, mean, median, linear interpolation, spline interpolation, multiple imputations, etc. I made the wrong choice of going with replacing the null values with the mean value. I realized this too late and it was too hard for me to fix this as it would have taken forever with my computer problems. Using the mean is very sensitive to outliers and is only good for MCAR data not time-series like ours. I should have used linear interpolation as its best for time-series data and approximates the null value by taking the value before and after it. This was a huge downfall in my model, but at least I found and learned from it. I commented the code as well.

Moving forward, I filled out the draft column and checked it to see if no null entries are present. I also made some simple plots and heatmaps, to visualize my data, but these weren't that helpful/ readable. So with this conclude my analysis and preprocessing of the data.

4 Building the model:

I first define X and y. "y" is rot-realised, which is what we aim to predict. "X" has all other parameters besides the date, and trip-id as these will not influence the prediction and are only time-stamps and id.

4.1 Splitting the data:

I first went the basic way by importing train-test-split from sklearn.model-selection and making my X-train, y-train, X-test,y-test. However, I realized that this is not at all optimal for time series data and that this import from sklearn is randomized which will hurt the accuracy of our model. This was my first time working with splitting test/train time-series data so I spent quite some time on this and debugging. So I used the the TimeSeriesSplit import from sklearn and initialized my n-splits=2. I was getting errors when I would use this in the model, so I manually split the test and train data using the trip-id as the heuristic. The last trips are test data and the rest are training data. I made this decision by analyzing the entries for each trip and allocating test and train sizes accordingly. I then split columns between X-train, y-train, X-test, and y-test just like how we did before for X and y.

5 Building, Training, and Predicting:

5.1 Linear-regression model and its evaluation

I first built a simple linear regression model. In linear regression, the model has to find the linear relationship of it with the dependent variable. The algorithm's aim is to find the best fit linear line such that the error is minimum for the best values of intercept. I then fitted the model and trained it. The MSE of this model was pretty high. I tried using the accuracy-score import but that is only for classification models and not regression models. So I used the R^2 score score metric for regression instead. The score was very low and did not perform well. I think the main problem with linear regression is autocorrelation. If autocorrelation residues are present, linear regression would not be able to catch all patterns in the time-series data.

dt	x	y	speed	u	v	cog	heading	lat	lon	draft	rudder	rot_realized	Predicted rot_realised
2021-07-.037575+00:00	3920563.738	3163362.399	0.051444	0.050473	-0.009949	166.416029	177.567353	51.438649	4.234213	1.812151	2.9	3.3	0.737105
2021-07-.349613+00:00	3920563.747	3163362.362	0.051444	0.050650	-0.009004	167.487329	177.567353	51.438649	4.234213	1.812151	3.2	3.3	0.742019
2021-07-.846008+00:00	3920563.745	3163362.343	0.051444	0.050590	-0.009338	167.116874	177.575190	51.438649	4.234213	1.812151	3.2	3.3	0.742923
2021-07-.347429+00:00	3920563.743	3163362.306	0.051444	0.051087	-0.006057	170.821657	177.582739	51.438648	4.234213	1.812151	3.2	3.3	0.734303
2021-07-.858551+00:00	3920563.741	3163362.288	0.056589	0.056427	-0.004280	173.245044	177.582739	51.438648	4.234213	1.812151	3.2	3.5	0.728296

Figure 1: Resultant table with predicted rot-realized by linear regression

5.2 Gradient Boosting Regression model and its evaluation:

Considering the down downfalls of linear regression and its low score, I thought using GBR would be a nice idea. Although the foundation of GBR is based on linear regression is much better than AdaBoost as its trees are constrained hence weak learners will stay weak but the prediction of the trees will be added sequentially hence It can increase the performance of the model by reducing overfitting. I forgot to do this, but as I write this report I realized I could have added and tested with more parameters such as the learning-rate, etc, to get a better performing model. The MSE of the GBR model was the best of the three and its R^2 score was slightly better than linear regression.

dt	x	y	speed	u	v	cog	heading	lat	lon	draft	rudder	rot_realized	Predicted rot_realised
2021-07-.037575+00:00	3920563.738	3163362.399	0.051444	0.050473	-0.009949	166.416029	177.567353	51.438649	4.234213	1.812151	2.9	3.3	1.843553
2021-07-.349613+00:00	3920563.747	3163362.362	0.051444	0.050650	-0.009004	167.487329	177.567353	51.438649	4.234213	1.812151	3.2	3.3	1.843553
2021-07-.846008+00:00	3920563.745	3163362.343	0.051444	0.050590	-0.009338	167.116874	177.575190	51.438649	4.234213	1.812151	3.2	3.3	1.843553
2021-07-.347429+00:00	3920563.743	3163362.306	0.051444	0.051087	-0.006057	170.821657	177.582739	51.438648	4.234213	1.812151	3.2	3.3	1.843553
2021-07-.858551+00:00	3920563.741	3163362.288	0.056589	0.056427	-0.004280	173.245044	177.582739	51.438648	4.234213	1.812151	3.2	3.5	1.843553

Figure 2: Resultant table with predicted rot-realized by gradient boosting regression

5.3 Random Forest regression model and its evaluation.

The ensemble-based random forest regression works by "bagging". The above models are data-sensitive hence small changes can affect the performance of the model along with the tree structure. Their search depth is not fully used hence they lean towards local solutions rather than global ones. Random forest constructs numerous decision trees at training time and returns the mean for each. Hence it is not data-sensitive and results from each tree are independent of each other. I trained the model. Despite its MSE being more than gradient boosting regression its R^2 score was the best among the three. Hence I feel it was the best fit.

dt	x	y	speed	u	v	cog	heading	lat	lon	draft	rudder	rot_realized	Predicted rot_realised
2021-07-.037575+00:00	3920563.738	3163362.399	0.051444	0.050473	-0.009949	166.416029	177.567353	51.438649	4.234213	1.812151	2.9	3.3	1.843553
2021-07-.349613+00:00	3920563.747	3163362.362	0.051444	0.050650	-0.009004	167.487329	177.567353	51.438649	4.234213	1.812151	3.2	3.3	1.843553
2021-07-.846008+00:00	3920563.745	3163362.343	0.051444	0.050590	-0.009338	167.116874	177.575190	51.438649	4.234213	1.812151	3.2	3.3	1.843553
2021-07-.347429+00:00	3920563.743	3163362.306	0.051444	0.051087	-0.006057	170.821657	177.582739	51.438648	4.234213	1.812151	3.2	3.3	1.843553
2021-07-.858551+00:00	3920563.741	3163362.288	0.056589	0.056427	-0.004280	173.245044	177.582739	51.438648	4.234213	1.812151	3.2	3.5	1.843553

Figure 3: Resultant table with predicted rot-realized by random forest regression

6 Final Results

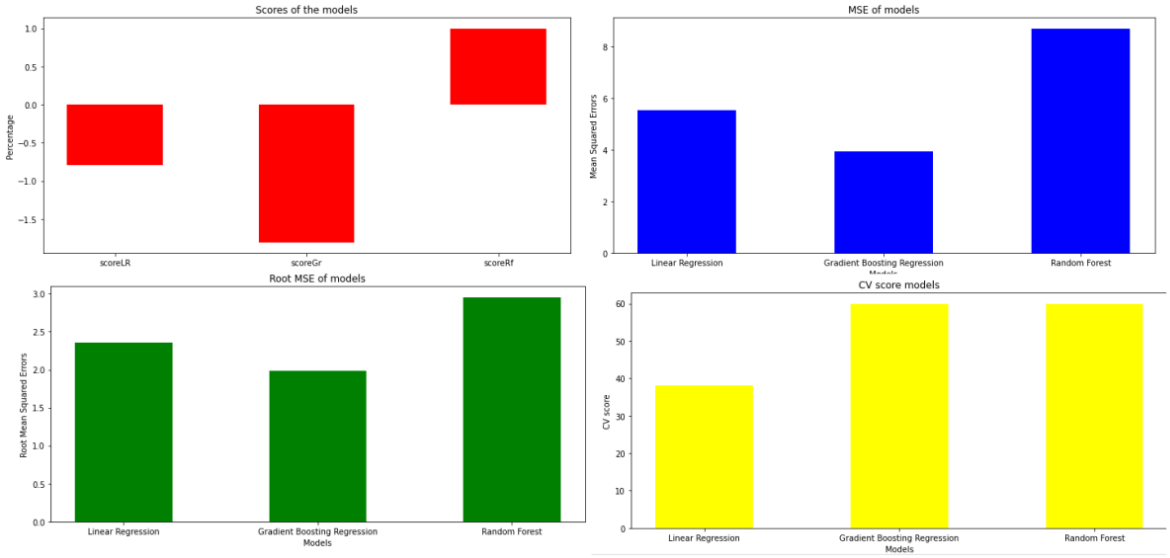

Figure 4: Evaluation of the three models.

7 Conclusion

It was interesting to work with shipping time-series data. I have never done this in the past. I was surprised by the volume of data from just one week of collection. Thank you for this opportunity.