Going green

Manuela Girotti

Université de Montréal, IFT 6390 - Introduction to Machine Learning, Final Project

Abstract

We study a yacht dataset with the help of four different regression algorithms in order to evaluate performance of sailing yachts. The goal is to produce a regression model that can predict a performance factor (residuary resistance) for a sailing boat.

This work is inspired by the ever-pressing issue of climate change and the need to adopt environmentally friendly solutions in our lives.

1 Overview

We consider a collection of data on the performance of sailboats and we want to analyze the data with the help of (at least) four different regression algorithms.

The prediction of residuary resistance (the output) is of great value at the initial design stage of sailing yachts, in order to evaluate the ship's performance and for estimating the propulsive power. Essential inputs are the hull geometry and the boat velocity.

The amount of available data is limited. In our analysis we will therefore give preference to algorithms that have low capacity or ensemble methods, in order to avoid overfitting: k nearest neighbours, linear regression and boosting method with decision trees. Naïve Bayes Regressor was also an option, but it has been proven to be less effective than other low-capacity methods (see [2]).

To validate our outcomes from the training set, we will use a 5-fold cross validation (see [3]. [4]). In the last part of the work, we also analyze the datasets with a fully connected neural network (which has potentially very high capacity) in order to have a more global understanding of the validity of our approach and results.

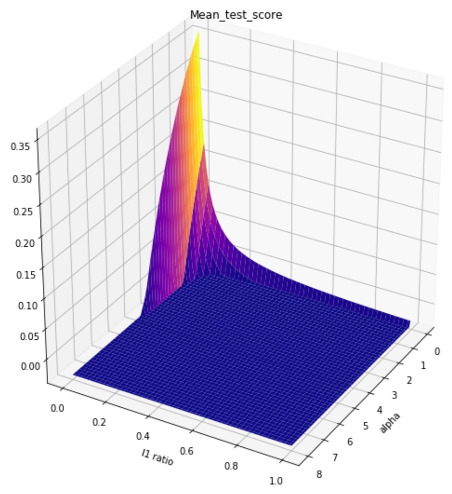
1.1 Dataset

Yacht Hydrodynamics dataset [1]: the dataset comprises 308 samples, collected at the Delft Ship Hydromechanics Laboratory. These experiments include 22 different hull forms, derived from a parent form closely related to the "Standfast 43" designed by Frans Maas. Each sample has the following features:

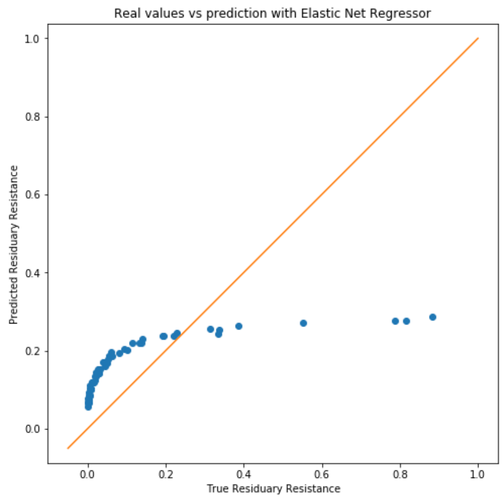
- x_1 = longitudinal position of the center of buoyancy,
- x_2 = prismatic coefficient,
- x_3 = length-displacement ratio,
- x_4 = beam-draught ratio,
- x_5 = length-beam ratio,
- x_6 = Froude number,
- y = residuary resistance[1] per unit weight of displacement.

We refer to Appendix B for the analysis of the boxplots of each feature.

[1]The Froude number Fr is a dimensionless value that describes different flow regimes of open channel flow. The Froude number is a ratio of inertial and gravitational forces.



(a) Grid search of the best value of α and L^1-ratio.



(b) True values vs Predictions.

Figure 1: Elastic Net algorithm.

1.2 Preprocessing and Evaluation

We first rescale the samples using `MaxAbsScaler` (see [5]). This will scale and translate each feature individually such that the maximal absolute value of each feature will be equal to 1. Since the quantities under exam are physical quantities, we compute and store also the inverse scaling transformation, so that, when calculating statistical (error) quantities, we can observe their value in the true scale.

Next, we split the datasets into two parts: about 80% of the samples will be part of the training (and validation) set and 20% will be used for testing. The statistical scores that we will consider in order to assess the validity of each algorithm are the Coefficient of Determination (R^2), the Mean Square Error (MSE) and the Mean Absolute Error (MAE).

2 Regression algorithms

Linear Regression: in a linear regression algorithm with elastic net the cost function is the standard mean square error, equipped with a regularizer of the form

$$J(D_N, \boldsymbol{\theta}) = \frac{1}{2N}\sum_{i=1}^{N}(y_i - \mathbf{W}\mathbf{x}_i)^2 + \alpha\lambda_{12}\left\|\mathbf{W}\right\|_1 + \frac{\alpha}{2}(1-\lambda_{12})\left\|\mathbf{W}\right\|_2^2$$

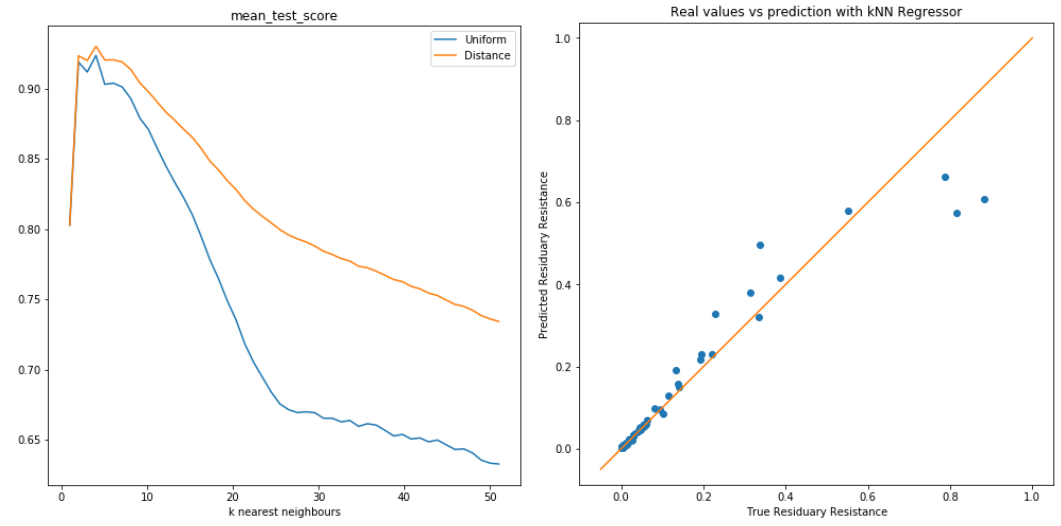
where $\lambda_{12} \in [0,1]$ is the ratio of importance given to the L^1 norm of the weights with respect to the L^2 norm (in particular with $\lambda_{12} = 0$ we have ridge regularization and with $\lambda_{12} = 1$ we have a LASSO regularization.

We explore with `GridSearchCV` the different possible values of regularization hyperparameter $\alpha \in [0.1, 10]$ and $\lambda_{12} \in [0,1]$ to identify the best performing one. With the possible values of $\alpha \in [0,10]$ (100 sample points) and $\lambda_{12} = [0,1]$ (100 sample points), we obtained that the best performance is obtained for $\alpha = 0.1$ and $\lambda_{12} = 0$. See Figure 1.

k **Nearest Neighbours:** in a k-NN algorithm, the prediction function is the weighted mean of the target values of the training points.

$$f(\mathbf{x}) = \frac{1}{\sum_{i=1}^{n} w_i}\sum_{i=1}^{n} w_i y_i \tag{1}$$

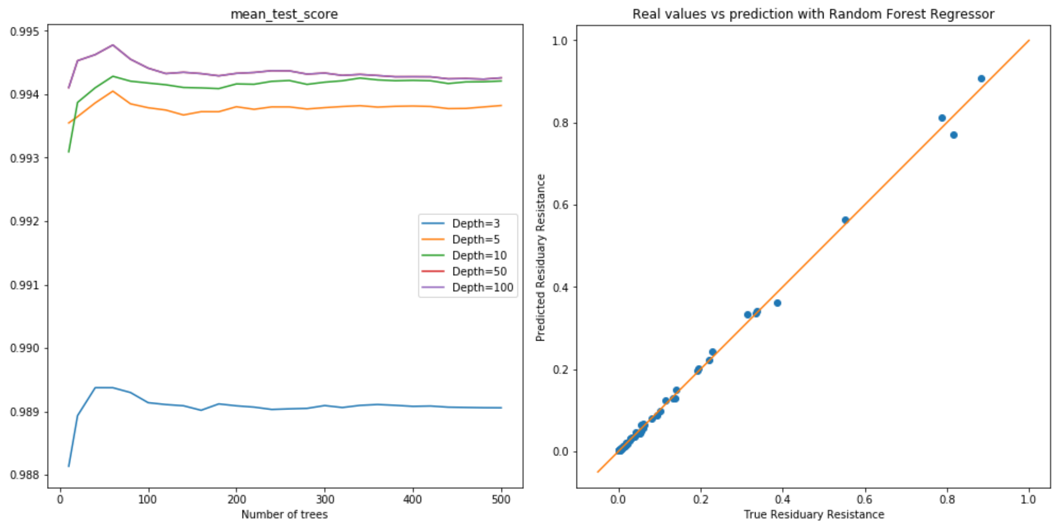
where n is the number of training samples, $w_i = w(\mathbf{x}_i)$ is the weight of each sample.



(a) Grid search for the value of k and type of weight.

(b) True values vs Predictions.

Figure 2: k-NN algorithm.



(a) Different random forest options.

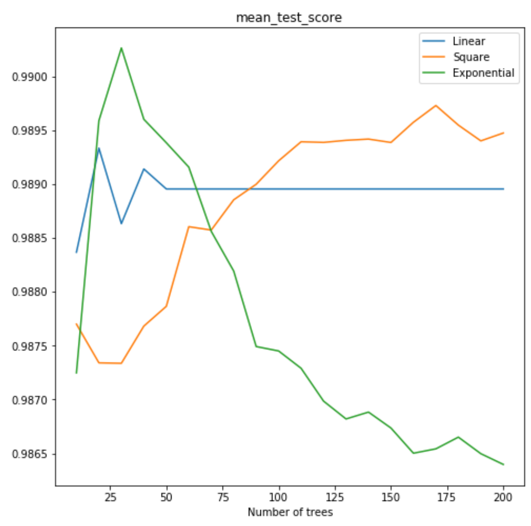
(b) True values vs predictions.

Figure 3: Random Forest algorithm.

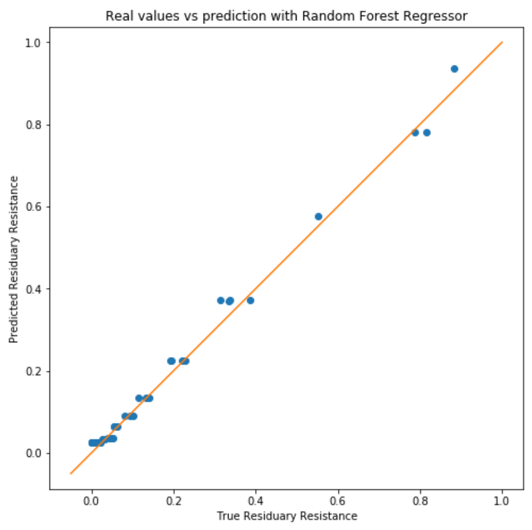
The standard k-NN algorithm considers each training sample to have weight equal to 1, if it is part of the k closest neighbours to the test point, and equal to 0 otherwise (`uniform` weight model). However, it is possible to use a more general version: the weights depend not only on the membership to the k closest neighbours, but also on their inverse (Euclidean) distance from the test point.

The best estimator for the Yacht dataset with a k-NN algorithm was a regressor with inverse distance as weights and $k = 4$ neighbours considered for each test point. These optimal hyperparameters were identitfied thanks to a a grid search (`GridSearchCV` with 5-fold cross-validation) among the following possible values: $k \in \{1, 2, \dots, 50\}$ and $\{w_i\} =$ `uniform` or `distance`. See Figure 2.

Ensemble Methods: We also analyze the dataset with the following two popular Ensemble methods: Random Forest Regressor and AdaBoost Regressor with Decision Tree as base estimator.



(a) Different loss functions.



(b) True values vs predictions.

Figure 4: AdaBoost algorithm.

Random Forest Regressor: we inspect the best number of estimators (Decision Trees) and the best depth of the estimators with the following values:

no. estimators $\in \{10, 60, 100, 150, 200, 250, 300\}$ and maximum depth $\in \{3, 5, 10, 50, 100\}$

where for each of the (weak) estimators we subsample the dataset (bootstrapping). We are not subsampling the number of features for each estimator because it is already a small number ($d = 6$), therefore reducing even further the number of features to be considered wouldn't improve the performance.

The best regressor is obtained with 60 trees with maximal depth equal to 50: $R^2 = 0.997785$ (this score is higher than the score obtained with XGBoost), $MSE = 8.444033e - 05$. See Figure 3.

AdaBoost Regressor: AdaBoost is a meta-estimator that begins by fitting a (shallow) Decision Tree regressor on the original dataset and then fits additional copies of the regressor on the same dataset with weights of instances adjusted according to the error of the current prediction. The main hyperparameters of the algorithm are the number of estimators, the learning rate and the loss function to use when updating the weights (linear, quadratic or exponential):

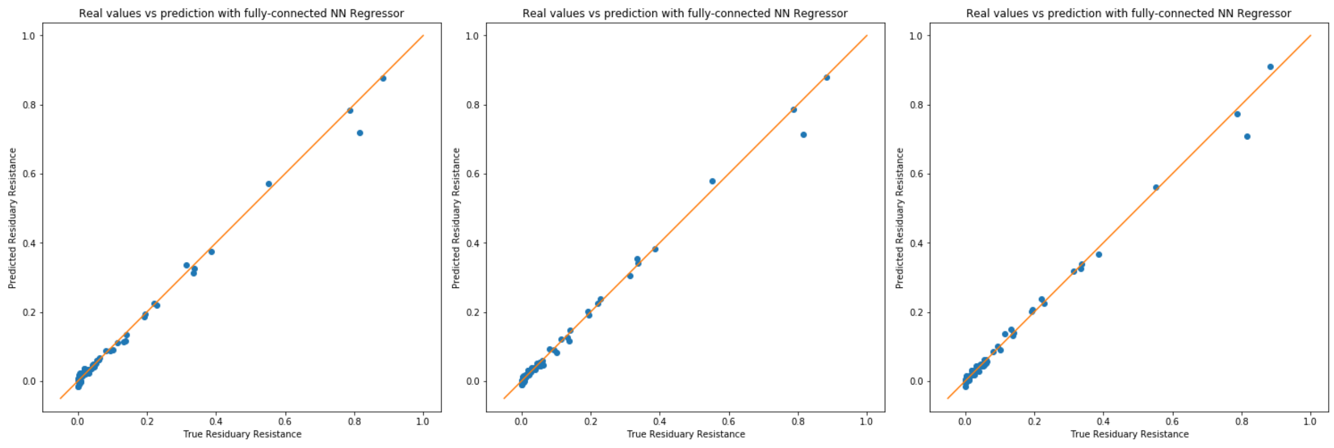
no. estimators $\in \{10, 50, 100, 150, 200\}$ learning rate $\in (0.1, 1.5)$.

In particular, the learning rate shrinks the contribution of each regressor: the smaller the learning rate, the more weak learners need to be taken into account.

The best regressor is obtained with 30 shallow decision trees (default depth = 3), exponential loss function and learning rate 1.03: $R^2 = 0.990912$, $MSE = 0.000346$. See Figure 4.

Neural Networks: We implement different architectures of a Multi-Layer Perceptron, to identify the one that is best suited for regression with a small dataset. Our general guidelines are the following: deeper and wider NNs have more capacity, therefore they are more likely to overfit; adding more hidden layers to a network means to allow the network to compute more complex features; adding more neurons to a layer means to create more (same-complexity) features; layers are usually kept narrower towards the end of the network, because complex features carry more information than simple ones and one doesn't usually need as many.

For the yacht dataset, our first baseline is inspired by the paper [6] where they analyzed the given data and additional artificially generated data with a NN with one hidden layer (with 9 neurons). Next, we increment the number of layers (up to three) and the number of neurons and compare the results. For the training, we use the Broyden-Fletcher-Goldfarb-Shanno algorithm (see Appendix A for experiments run with SGD).



(a) 1-layer NN. (b) 2-layer NN. (c) 3-layer NN.

Figure 5: Results for the Yacht Dataset.

We explore the possibility of adding an L^2-regularization term and we carefully select the best suited activation function for our data (σ, $\tanh$ or ReLU). We perform a grid search with hidden layers equal to `9`, `30` or `100` in the one-layer case ; `(9,5)`, `(30,20)` or `(100,50)` for the two-layer case, and `(9,5,3)`, `(30,20,10)` or `(100,50,10)` for the three-layer case; in all three cases, the regularization term could have had values between 0 and 1.

The performances (R^2 score) are quite comparable and in order to avoid overfitting, we will retain the first architecture and report it in the final table of results.The results are collected in Table 6.

NN - Layers	Best architecture	Activation funct.	Regularization	R^2
1 layer	6:100:1	ReLU	0.0	0.993380
2 layers	6:100:50:1	ReLU	0.0	0.993580
3 layers	6:30:20:10:1	Tanh	0.0	0.993163

Figure 6: Table of results of the NN model.

3 Discussion of the results

We collect the best results in Table 7. The scores are all calculated on the comparison between the prediction given by the trained algorithm and the true value of the output from the 20% of the datasets that we saved as "test set" (see Section 1.2).

Regression algorithm	R^2	MSE (true scale)	MAE (true scale)
k-NN	0.919128	12.016490	1.371772
Lin. Regr.	0.336974	98.516976	7.569683
Decision Tree (RForest)	**0.997785**	**0.329001**	**0.315433**
Decision Tree (AdaBoost)	0.990913	1.350214	0.991270
NN	0.993380	0.983528	0.584594

Figure 7: Table of results.

Despite the very small capacity of the k-NN model, its performance is quite remarkable and can be easily compared with more sophisticated models like Random Forests and NN. However, given the high dimension of the feature spaces ((we have 308 sample points in $\mathbb{R}^6$), this model will most certainly suffer from the curse of dimensionality and it will probably perform badly on new unseen examples if they consistently differ from the samples collected in the datasets.

The outcome of the linear regression model for the yacht dataset is quite disappointing: a reason could be that the features may be highly correlated and a linear predictor doesn't quite catch the underlying structure of the data. It may be tempting to preprocess the data by applying a suitable feature map or try to use the kernel trick (Kernelized Ridge Regression or similar algorithms).

While performing the training on the Elastic Net algorithm, we had a warning message saying that the gradient descent did not converge, despite the big number of iterations (about 1 million). This also explains the poor performance of the algorithm and it hints that it is not the best choice for analyzing our datasets.

Finally, the Neural Network algorithm performs very well, as expected. However given the small amount of data and the powerfulness of a neural network, this algorithm is the more likely to overfit with respect to the other algorithms considered. On the other hand, boosting algorithms are notably very robust to overfit and the best performance was achieved with a Random Forest algorithm.

4 Conclusions and further developments

The main difficulty in the study of these datasets is given by the reduced amount of available data. This carries significant implications in the selection of the most appropriate model to work with, in order to avoid overfitting (if the model has too high capacity, like neural networks) or underfitting (if the model has very low capacity, like linear regression).

A very attentive examination of different statistical error measures (R^2 score, MSE, etc.) is also required to correctly interpret the predictions given by the model. The need for a larger dataset can be met by using data augmentation techniques (like SMOTE or Modified-SMOTE) to generate extra synthetic data to complement with the available ones. We could additionally produce confidence intervals (as opposed to point estimates) to have more reliable results.

A Neural network experiments with stochastic gradient descent for the Yacht Dataset

To fully understand the outcome of the Neural Network experiments, we also ran some experiments using stochastic gradient descent instead of the BFGS method. The results that we obtained are substantially worse and, indeed, for small datasets BFGS generally converges faster and performs better than SGD.

We ran the following three experiments with grid search over different values of the number of neurons, regularization parameter, step size, activation function and possibility of early stopping:

1. one hidden layer, backpropagation with SGD;

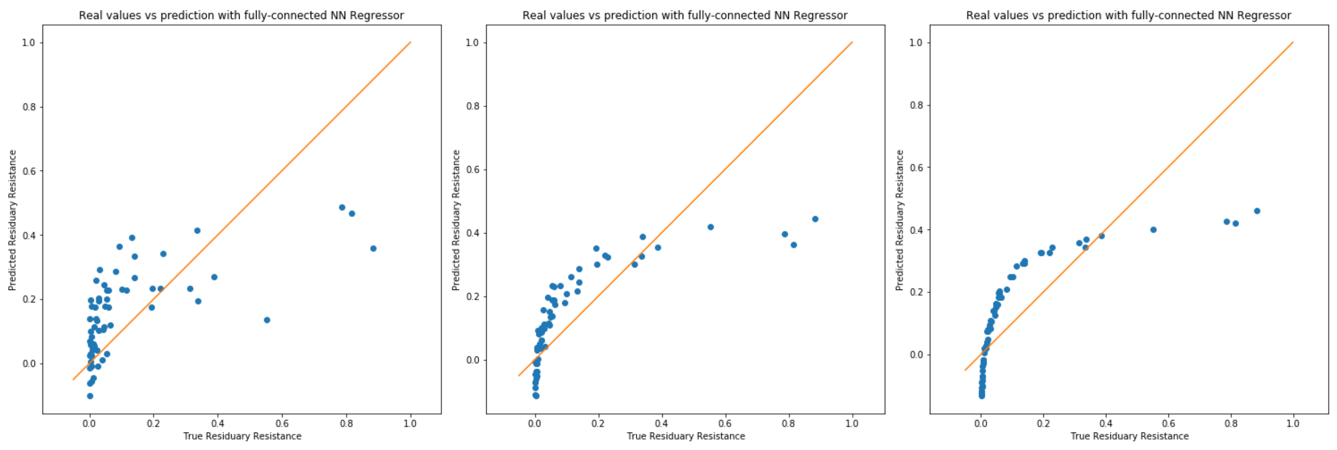
```
grid_parameters_MLP_regression =
        {'hidden_layer_sizes':[(9,),(30,),(100,)],
        'activation':['logistic', 'relu','tanh'],
        'learning_rate':['constant', 'adaptive'],
        'alpha': np.linspace(0,1.0,50),
        'early_stopping':[False,True]}
```

2. two hidden layers, back propagation with SGD;

```
grid_parameters_MLP_regression =
        {'hidden_layer_sizes':[(9,50),(30,50),(100,50)],
        'activation':['logistic', 'relu','tanh'],
        'learning_rate':['constant', 'adaptive'],
        'alpha': np.linspace(0,1.0,10),
        'early_stopping':[False,True]}
```

3. three hidden layers, back propagation with SGD;

```
grid_parameters_MLP_regression =
        {'hidden_layer_sizes':[(9,5,3),(30,20,10),(100,50,10)],
```



(a) 1-layer NN. (b) 2-layer NN. (c) 3-layer NN.

Figure A.1: Results of the NN model on the Yacht Dataset with SGD.

```
'activation':['logistic', 'relu','tanh'],
'learning_rate':['constant', 'adaptive'],
'alpha': np.linspace(0,1.0,10),
'early_stopping':[False,True]}
```

The results are shown in Figure A.1 and collected in the Table A.2. In Figure A.1 we can clearly detect a trend that becomes more and more defined by increasing the number of hidden layers. We suspect that with the help of additional Physics (hydrodynamic) knowledge we could manually (and mindfully) correct the algorithm and tailor it to the specific prediction task for a yacht dataset: this translates into finding an explicit feature map to apply to the samples in order to better understand their topology and predict the outcome with higher accuracy.

NN (yacht)	backprop.	Activation funct.	Regularizer (α)	Early Stopping	Step size	R^2
1 layer	SGD	Relu	0.1	False	adaptive	0.346087
2 layers	SGD	Tanh	0.0	True	adaptive	0.567313
3 layers	SGD	Tanh	1.0	False	adaptive	0.561937

Figure A.2: Table of results of the NN model on the Yacht Dataset with SGD.

It is interesting to notice that the NNs with 2 or 3 layers perform better than the NN with only one hidden layer, but the architecture with 2 layers required early stopping during the training in order to achieve the best result (unlike the other cases).

B Analysis of the data landscape

Observing the boxplots (Figure B.1) of the yacht dataset, we can notice that the features have relatively concentrated values around their median, but half of the features (length-beam ratio, length-displacement ratio, prismatic coefficient) has values that are extremely skew. The outcome (residuary resistance) also shows a good amount of (one-sided) outliers.

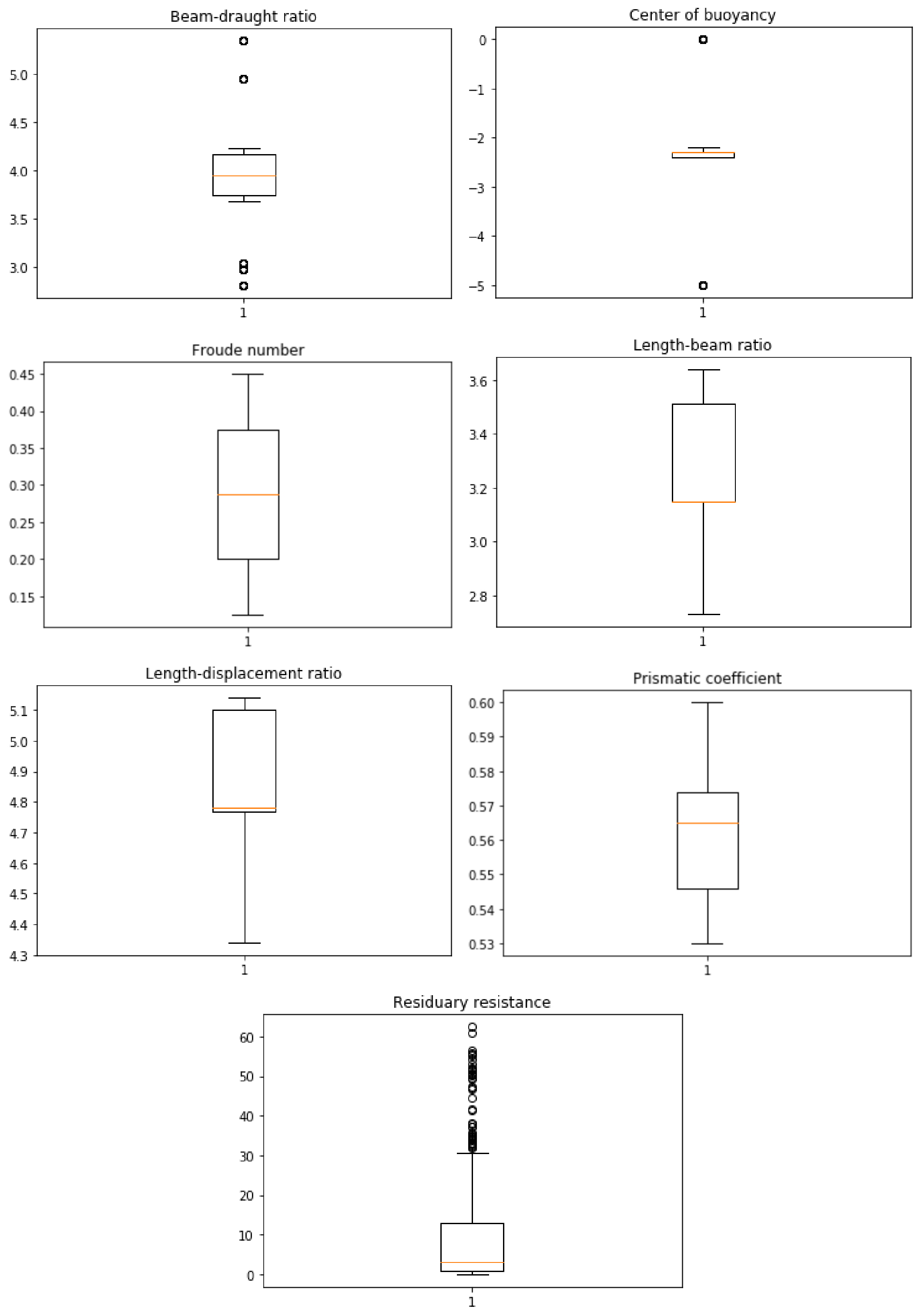


Figure B.1: Boxplots of the Yacht Hydrodynamics dataset.

References

[1] Yacht Hydrodynamics Data Set, Dua, D. and Graff, C. (2019). UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Science. http://archive.ics.uci.edu/ml/datasets/Yacht+Hydrodynamics

[2] Frank, E., Trigg, L., Holmes, G., and Witten, I., and Aha, W., *Naive Bayes for Regression*, Machine Learning, 2001.

[3] Breiman, L., and Spector, P., *Submodel selection and evaluation in regression: The X-random case*, International Statistical Review, 60, 291-319, 1992.

[4] Kohavi, R., *A study of cross-validation and bootstrap for accuracy estimation and model selection*, International Joint Conference on Artificial Intelligence (IJCAI), 1995, http://robotics.stanford.edu/users/ronnyk/.

[5] Wenkel, S., *Revisiting Machine Learning Datasets - Yacht Hydrodynamics*, 2018, https://www.simonwenkel.com/2018/09/08/revisiting-ml-datasets-yacht-hydrodynamics.html#exploring-the-dataset-and-preprocessing.

[6] Ortigosa, I., López R., and García. J., *A neural networks approach to residuary resistance of sailing yachts prediction*. In Proceedings of the International Conference on Marine Engineering MARINE 2007, 2007.

[7] Pedregosa et al., *Scikit-learn: Machine Learning in Python*, JMLR 12, pp. 2825-2830, 2011.