Using Machine Learning For Yacht Hydrodynamics

David Llewellyn Smith

December 6, 2024

1 Introduction

Being able to predict performance in the design stage is extremely important because it allows engineers to optimize design and performance before any physical prototyping has to occur. In sailing yachts, performance can be measured by the residuary resistance per unit weight of displacement, which measures how much resistance is created by the shape of the hull. Therefore, minimizing that resistance is an important goal for yacht designers, and that is simplified a lot by being able to predict that performance from design specifications.

The Delft data set, 308 experiments performed at the Delft Ships Hydrodmechanics Laboratory in the Netherlands, collects results for 22 different hull forms traveling at different speeds. This allows us to use supervised machine learning to predict the residuary resistance from the other variables. We used a ridge regression approach and a neural network regression approach. We concluded that while both were effective, the neural network was far more effective, and almost perfectly predicted the measured resistances. [1]

2 Data

The features of interest in the dataset are:

1. The longitudinal position of the center of buoyancy (LCB), which is how far in front or behind of the middle of the ship is located the center of buoyancy as a fraction of the length of the ship.
2. The prismatic coefficient (Cp), which measures how sleek/square a boat is.
3. The length-displacement ratio (DLR), which measures how heavy a boat and is defined by

$$DLR = \frac{displacement(\text{lb})/2240}{(0.01 \cdot LWL(\text{ft}))^3} \tag{1}$$

   where LWL is the length of the boat at the waterline.
4. The beam-draught ratio (B/D), which is the ratio between the width and the depth of the ship.
5. The length-beam ratio (L/B), which is the ratio between the length and width of the ship.
6. The Froude number, which is a dimensionless ratio equal to $\frac{u}{\sqrt{gL}}$, which tells us how fast the ship is moving.

The first five features are design parameters of a ship, while the last one is a measure of velocity. We are hoping to use them to predict the residuary resistance per unit weight of displacement, which we expect to only depend on ship design and speed. The dataset was sourced from the UC Irvine Machine Learning Repository, and was pre-processed. Every variable is non-dimensional.

The 22 different designs, which were tested at Froude numbers between 0.125 and 0.450 with increments 0.025, are listed in Table 1.

As an example, Figure 1 shows the resistance versus the Froude number for the first of these designs. It exhibits a clear exponential relationship, and we can assume features of that function are determined by the design specifications.

LCB	Cp	DLR	B/D	L/B
-2.3	0.568	4.78	3.99	3.17
-2.3	0.569	4.78	3.04	3.64
-2.3	0.565	4.78	5.35	2.76
-2.3	0.564	5.10	3.95	3.53
-2.4	0.574	4.36	3.96	2.76
-2.4	0.568	4.34	2.98	3.15
-2.3	0.562	5.14	4.95	3.17
-2.4	0.585	4.78	3.84	3.32
-2.2	0.546	4.78	4.13	3.07
0.0	0.565	4.77	3.99	3.15
-5.0	0.565	4.77	3.99	3.15
0.0	0.565	5.10	3.94	3.51
-5.0	0.565	5.10	3.94	3.51
-2.3	0.530	5.11	3.69	3.51
-2.3	0.530	4.76	3.68	3.16
-2.3	0.530	4.34	2.81	3.15
0.0	0.600	4.78	4.24	3.15
-5.0	0.600	4.78	4.24	3.15
0.0	0.530	4.78	3.75	3.15
-5.0	0.530	4.78	3.75	3.15
-2.3	0.600	5.10	4.17	3.51
-2.3	0.600	4.34	4.23	2.73

Table 1: Design Specifications used in the data set

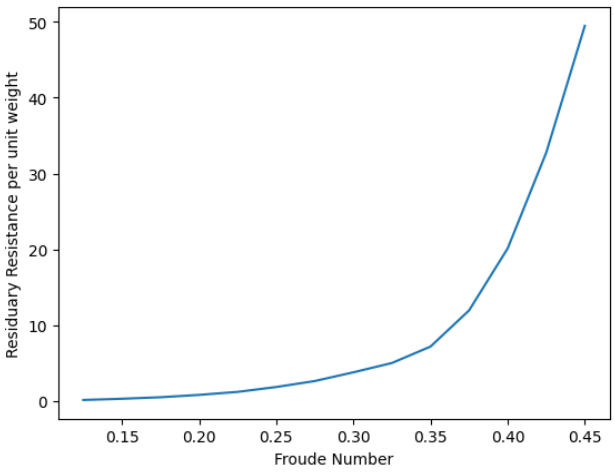

Figure 1: Resistance vs Froude number

3 Modelling

Following the flowchart provided by Professor Bortnik [2], this problem involves more than 50 samples, predicts a quantity, has less than 100 thousand samples, and uses a lot of features, so we decided to use Ridge Regression. That was implemented using the following code, calculating the root-mean square error along the way.

```
from sklearn.linear_model import Ridge
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
RR = Pipeline([('scaler', StandardScaler()),('ridge',Ridge(alpha=0.01,positive = True))])
RR.fit(x_train,Y_train)
Y_pred = RR.predict(x_test)
Y_pred_non_negative = np.clip(Y_pred, a_min=0, a_max=None)
print("RMSE for linear regression:", np.sqrt(np.mean((Y_test-Y_pred_non_negative)**2)))
```

Unsatisfied with the quality of the results, we then decided to try a neural network, which performed better. That was implemented using this code:

```
from sklearn.neural_network import MLPRegressor
neuralpipe = Pipeline([('scaler', StandardScaler()), ('MLP', MLPRegressor(
    hidden_layer_sizes=(30,30),max_iter=5000))])
neuralpipe.fit(x_train, Y_train)
y_pred_neural = neuralpipe.predict(x_test)
print("RMSE for Neural Network:", np.sqrt(np.mean((Y_test-y_pred_neural)**2)))
```

4 Results

Figure 2 compares the actual resistance values to those predicted by the Ridge Regression model. The x-axis is just the order of the experiments in the data file. The RMSE was found to be 8.414. The coefficient of determination between the predicted and actual values was found to be 0.3537, which is meaningful but not great.

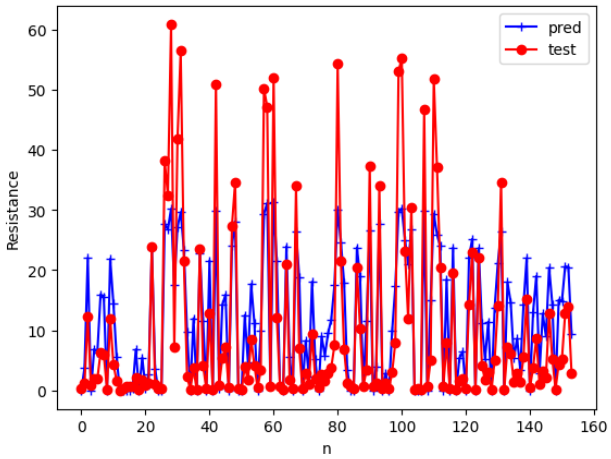

Figure 2: Test data for Ridge Regression

Figure 3 compares the measured resistance values to those predicted by the neural network. The RMSE was 1.836, a significant improvement from the regression model. The coefficient of determination was found to be 0.9838, which is exceptional. This suggests that the neural network was able to find non-linear relationships in the data that the ridge regression model could not find. Figure 4 shows a Regression Error

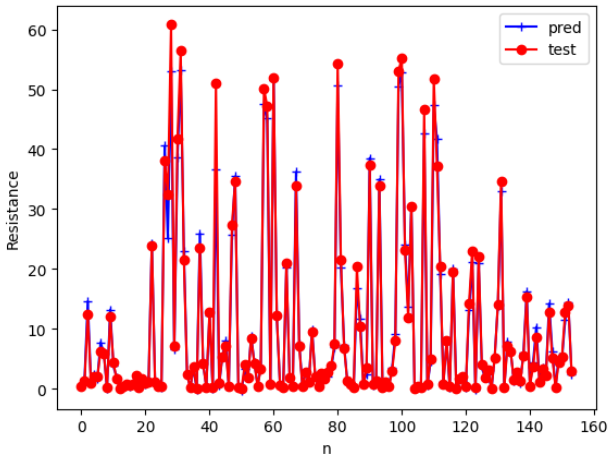

Figure 3: Test data for Neural Network

Characteristic curve for the two models used in this project, which allows us to evaluate the performance of the two models both on their own and against each other.

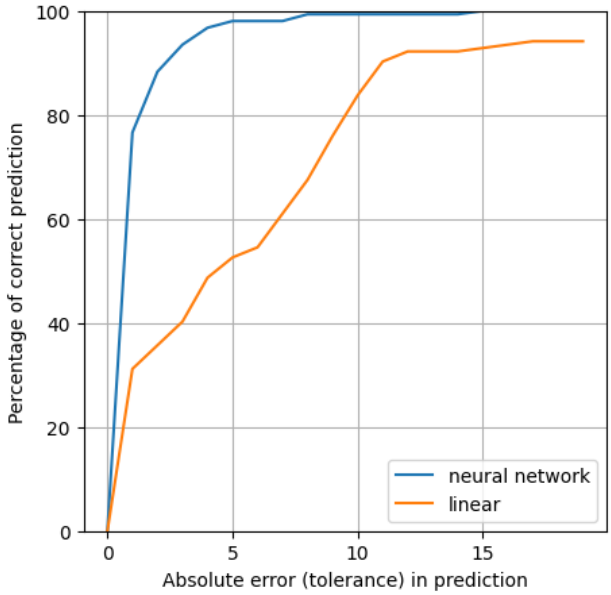

Figure 4: REC curve

4.1 Discussion

Figure 1, which shows the results from the ridge regression model, shows that the model overpredicts low resistances and underpredicts high resistances. That means the model is struggling with outlier values. Figure 2, for comparison, is far more accurate, using the MLP Regressor model. The blue crosses representing the predicted values are for the most part hidden by the actual measured values, which means the model performs extremely well. From Figure 3, we see that the neural network is within 95% accuracy for an error less than 5. The linear model is still above 90% accuracy for an error below 15, but this is far less effective than the neural network.

4.2 Conclusion

In summary, machine learning methods were used to predict hydrodynamic performance for yachts. A ridge regression method and a neural network method were both used, with the neural network performing better. We can conclude that machine learning can be effectively used to model yacht hydrodynamics. This will allow yacht designers to model performance before any prototypes are manufactured. We can also conclude, unsurprisingly, that a neural network approach is far more accurate than a simple ridge regression model, although certainly more computationally expensive.

A future project could apply the models we've created to other datasets testing more hull designs in order to fully validate the models. It could also examine which of the variables included in the data set affect the resistance the most in order to optimize design decisions. Finally, it could also investigate the equations that govern the problem to compare these models to what a physics-based solution looks like.

5 References

[1] UC Irvine Machine Learning Repository. Yacht Hydrodynamics Data Set.
[2] Bortnik, Jacob. Atmospheric & Oceanic Sciences C111. UCLA, 2024.