KraftHack Hackathon
TEAM: CLOPEN

Tobias Opsahl, Sigurd Holmsen, Øystein Høistad Bruce

March 7 2022

UiO : Universitetet i Oslo

Department of Mathematics

Contents

Abstract

In this exploration we have analysed a data on a turbine in Kvilldal, and made predictions for the tensile on the bolts for the near future. The dataset is of highly correlated variables, such as how much power is produces and how open the valve is. We discover that when the valve is closed and re-opened, there is firstly a big dip in tensile on the bolts, but then it goes further up than before opening. We modelled the prediction with linear regression and boosting. At first our models was not able to predict this higher tensile after re-opening, but with feature engineering and linear constraint the predictions looks promising for the near future. However, after looking at the dataset for more historical data, we see that this increase after re-opening is not global. Over the seasons, the bolts gradually decrease, then increase. We do not know how to model this global changes over time, but we notice some patterns. Our prediction model does not consider the historical data, because the test-set we are predicting on is in the near future. Therefore, we over-fit on the test-set, but add some analysis for bigger timeframes.

1 Introduction to the problem

The main task is to get good predictions on the **prediction_input** (file), with the usage of both files **input_dataset-1** and **input_dataset-2**. The data sets are censored values from the Unit 4 in the hydropower plant Kvilldal. Both files contains the values we want to predict, which are the 6 Bolts tensile value (not included in the **prediction_input**). The tensile values are among the most important features in the data set, because these values can be used to get knowledge about the risk in the turbine.

All the mentioned data set are more or less connected in time. The order are **input_dataset-1**, **input_dataset-2** and **prediction_input** and contains about 6 months, 1 month and 8 days (respectively). In this regard, we are main focus will be to use the second data set in the model training. We are doing it this way, since the newer data are "closer" to the data we want to predict w.r.t. time and the features we want to predict.

2 Data set of interest (focus: high prediction score)

This section will explain the approach to reach our final model. In this section, and the model fitting, we will just use the parameters that is included in the data set we want to predict on, in addition to the parameters we want to predict (the 6 different bolt tensiles).

One of the first things we did was to get the following correlation matrix between all the selected features (created by **input_dataset-2**).

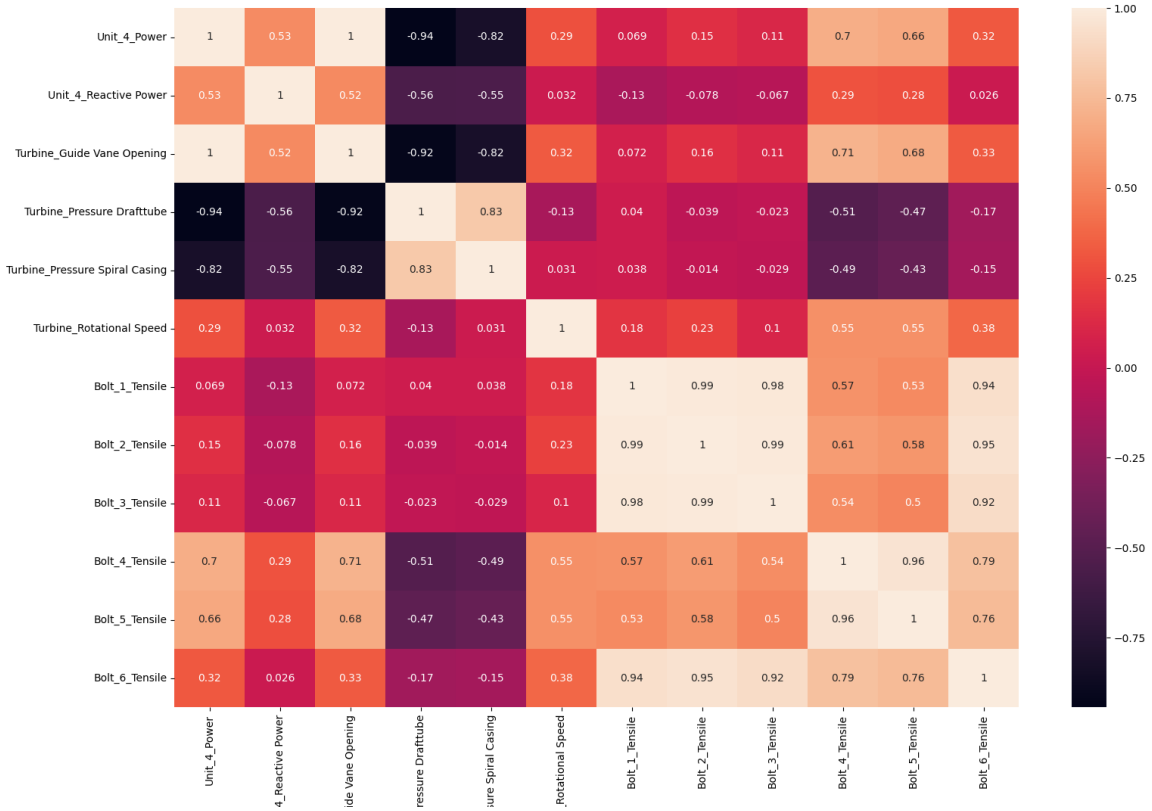

Figure 1: Correlation matrix of a cut in features. Keeping the feature that is "important" for the prediction

Figure 26, shows us some interesting correlation between some features. One observation is that the generated power and the turbine vane opening are 100% correlated. There seem to be a high correlation between the tensile in the Bolt group of 1, 2, 3 and 6, and equivalently for the Bolt group 4, 5. In this data set, there wasn't much correlation between the input data and the data we want to predict.

Next step was to plot the tensile values for a Bolt, over indexing - which in our case where datetime.

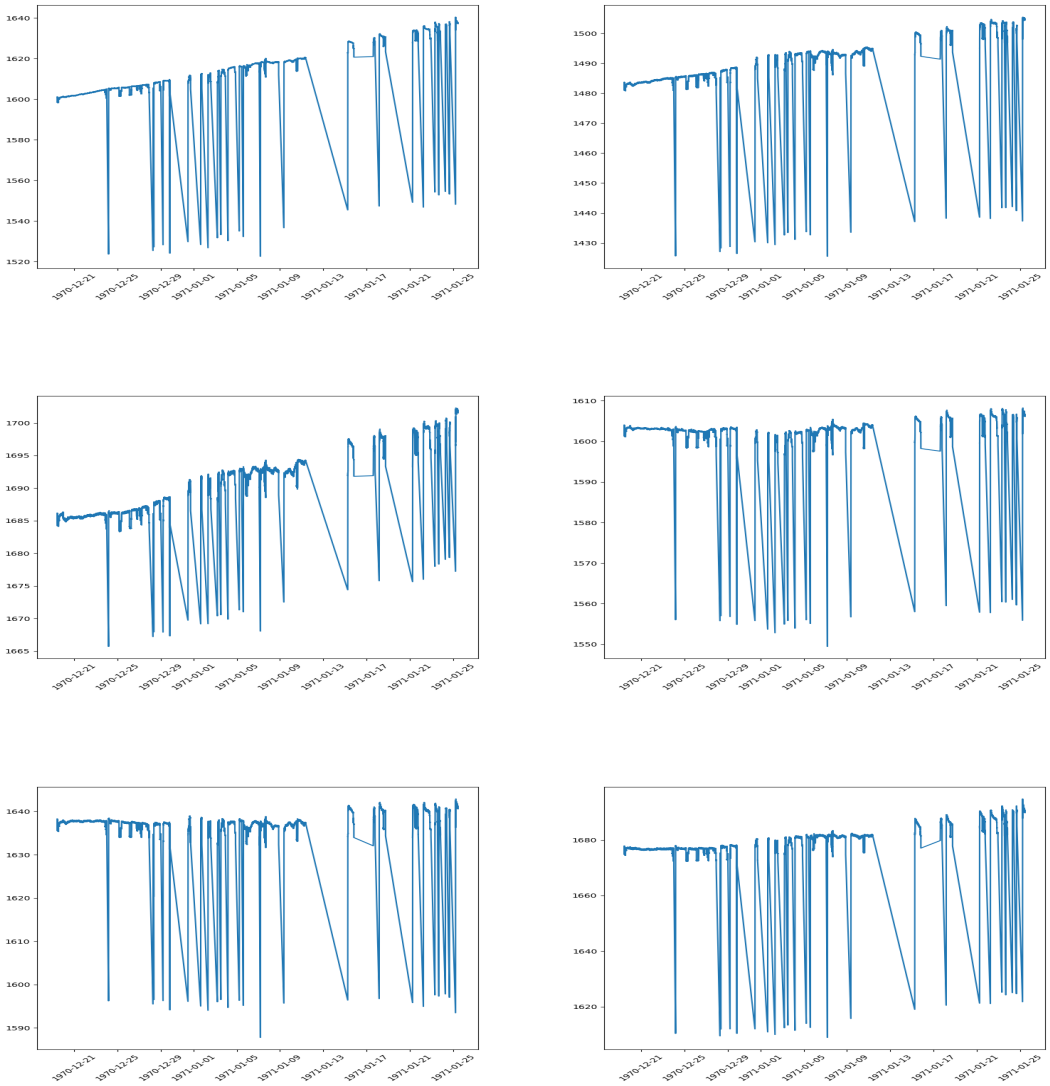

Figure 2: The tensile for different bolts over time
Row 1 Bolt 1, Bolt 2 — **Row 2** Bolt 3, Bolt 4 — **Row 3** Bolt 5, Bolt 6

The figure, gives us the insight that the Bolts tensile are very dependent on time (increasing in time). The value will sometimes drop, as you can see in the plots, which refer to the mode - if the turbine is on or off. By this, we included a new feature, which is the time with the unit of seconds from the first date in **input_dataset-2**.

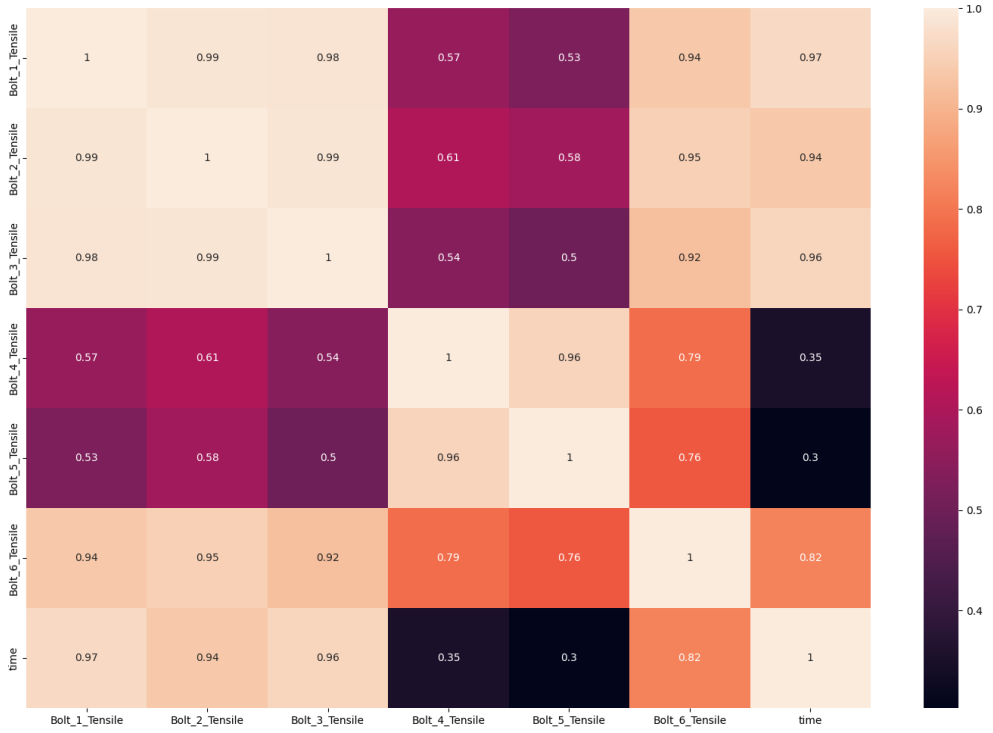

Figure 3: Correlation matrix between the bolts and the time

3 Making the models

Now it is time to actually make some models and predict the bolts tensiles. In this section we will gradually build up the models, to show some insights in the data explorations. Therefore, the first models presented perform badly, but we need to understand why in order to perform them.

In this section the *input_dataset2* is referenced to as the *training-dataset*, while the *prediction_input* is the test-set.

Strategy and assumptions

We chose to use two models during the process. One was a simple linear regression, and the other was gradient boosting. We included the linear model to have a reference point for how a simple model acted ([2]), which was able to make a decent fit in some instances. For the flexible model, we chose boosting because it is usually the model of choice for heterogenous data. In particular, we used the Light Gradient Boosting library, which is a module created by Microsoft that builds on top of ideas from XGBoost, but with leaf-wise tree growing and very fast training time ([1]).

Since much of the data is correlated, we are able to drop some features in the input dataset. We drop column 3, 4 and 6. This does not make a huge difference for the models. The code pipeline.py is generic enough so that this can easily be changed.

For the training-test split, we split the data horizontally in time, to mimic prediction dataset. Our prediction dataset takes time six days later than our train data, and is about 23% of the size, so the most realistic use the last 23% of the training-data as a test set. Since the data is highly correlated, doing a random split gives very much information of the test set. Having train data on both sides of a test observation makes it easy to make an extremely good fit, and we got the test-error extremely low this way. However, predicting further in time is a harder and more realistic task, so the horizontal split is better.

As we saw in the last section, bolt 1, 2 and 3 are highly correlated, and so are bolt 4 and 5. For simplicity, in the exploration stage of the model, we will only look at bolt 1. When we arrive at the final model, all of them will of course be included.

First Models

We then made train/test

Lets take a look at how the LightGBM model performed. We used all the explanatory variables, except the dates.

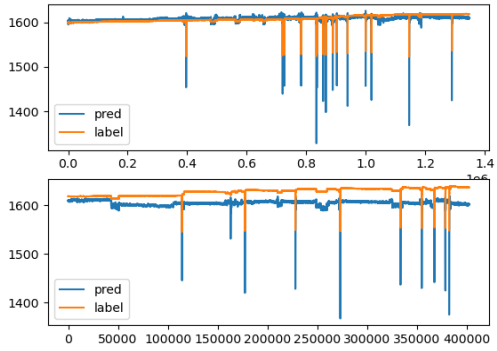

Figure 4: Linear model of bolt 1, without any time variable

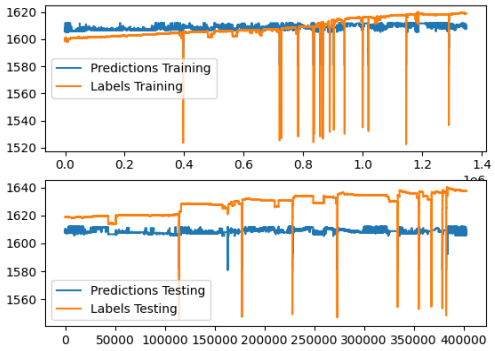

Figure 5: Boosting model of bolt 1, without any time variable.

As we see, the fit is not so good, and it misses substantially on the test set. By looking at the actual labels, we see that the values have a slight linear tendency over time. It therefore looks like time is a crucial feature. We start by using seconds after the first observation.

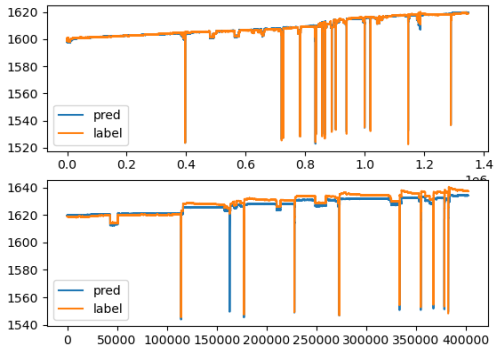

Figure 6: Linear model of bolt 1, with time variable

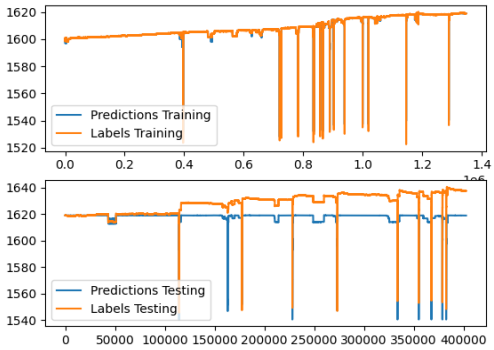

Figure 7: Boosting model of bolt 1, with time variable

Now the training fit is very close, but there is still a high test error. For some reason, the model fails to follow the linear trend in the testing data.

There is another observation about the labels which is important to note. When the tensiles have substantial dips (which happens after the valve is reopened), they go back to a little bit higher than normal, and stay there. This indicates that the stress on the bolts from reopening the valve does not degrade over time. Our model can not pick up on this effect, so we have to handle it somehow. We chose to feature engineer a cumulative effect of how long the valve was open, which changes each times it re-closes.

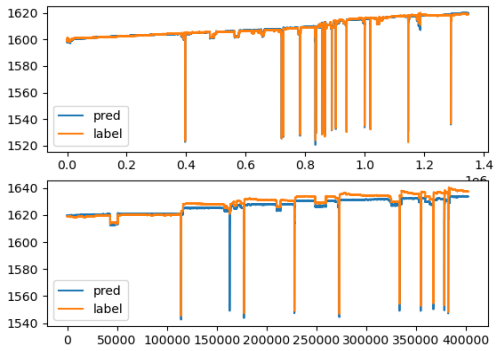

Figure 8: Linear model of bolt1, with time variable and feature engineering

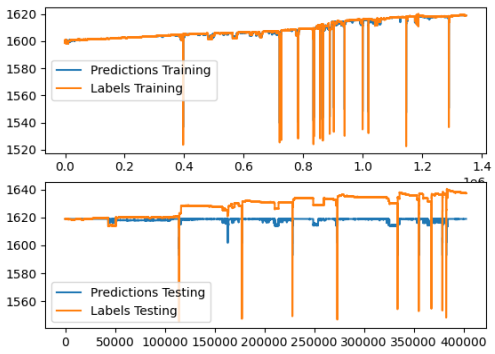

Figure 9: Boosting model of bolt1, with time variable and feature engineering

This greatly helps the linear model, but not the boosting.

Boosting with linear transformation

For the boosting model to fit good, we had to impose a strict linear constraint. For each of the bolts, we calculate the expected linear effect that time has on the tensile, which is present in our dataset. We then transform the response variable, and predict the flattened response. When we plot later, we transform it back again.

As we will soon see, this linear constraint makes our boosting-model perform surprisingly good, but there are some dangerous possible pitfalls here. This linear constraint is only assumed because

it is present in our training data, and might not be a global trend. Moreover, we will see that it is not linear in the bigger dataset in the next section. Therefore, this assumption may result in poor model predictions far into the future.

If the linear constraint assumption not valid, then why do we still chose to use it for prediction? This is because we know for a fact that the set we are being evaluated on is in a short period in the future right after the last time points in our training data set. Therefore, the local trends in our training-set probably extrapolates good onto the test-set, while the historical data would probably add noise. We therefore *deliberately over-fit on our test-set* to make some decent predictions, but we do not think this is a good all around strategy.

Enough writing, here are the plots.

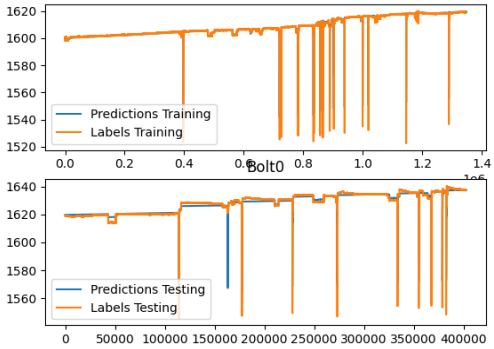

Figure 10: Bolt 1 of our final linear constrained boosting model (train-test plot)

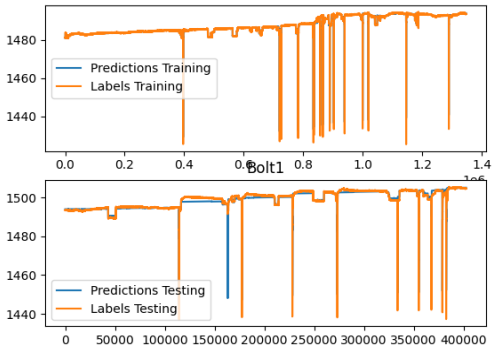

Figure 11: Bolt 2 of our final linear constrained boosting model (train-test plot)

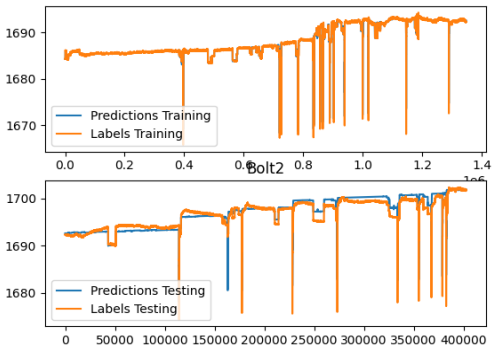

Figure 12: Bolt 3 of our final linear constrained boosting model (train-test plot)

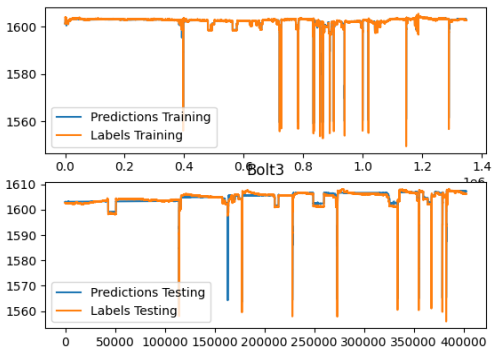

Figure 13: Bolt 4 of our final linear constrained boosting model (train-test plot)

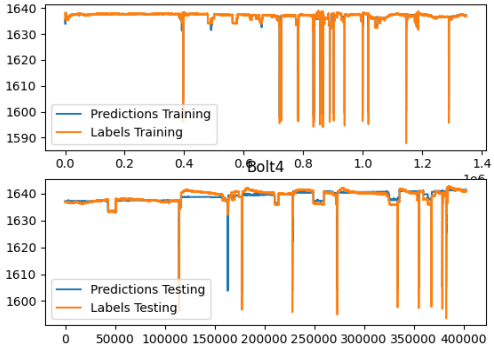

Figure 14: Bolt 5 of our final linear constrained boosting model (train-test plot)

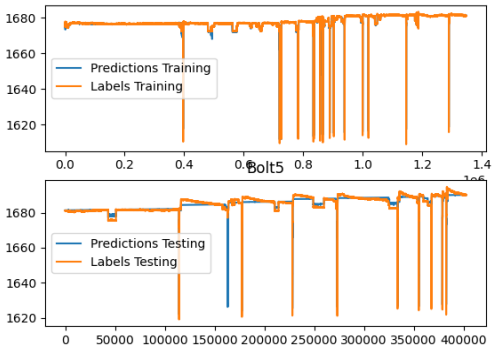

Figure 15: Bolt 6 of our final linear constrained boosting model (train-test plot)

The results look promising, although the model does not pick up some of the dips right away. We see that after turning on the valves, the tensiles jump slightly above where they were, which our model does not predicting accurately. However, it grows linearly over time, and therefore it quickly gets back on track.

Actual predictions

Now that we have figured out a descent way of predicting in the near future, let us predict on our actual test set. We fit our model on the whole training-set. The code for this (in pipeline.py) is pretty generic, so we only need to pass in the argument make_real_preds as True for the acutal test-dataset to be considered.

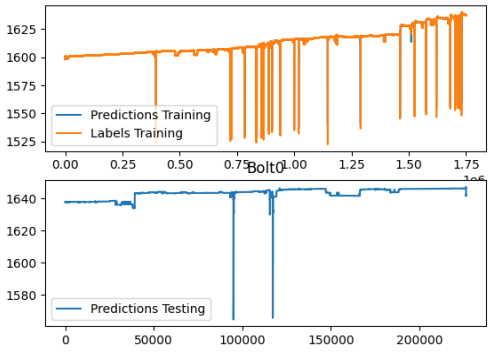

Figure 16: Bolt1 of our final linear constrained boosting model on the actual prediction dataset

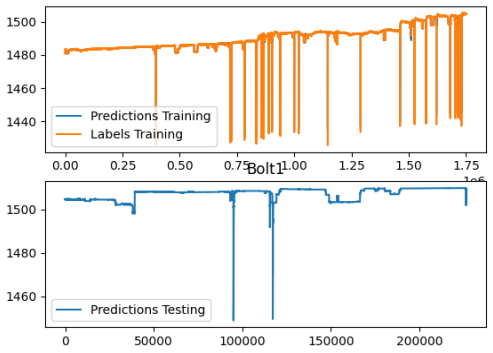

Figure 17: Bolt2 of our final linear constrained boosting model on the actual prediction dataset

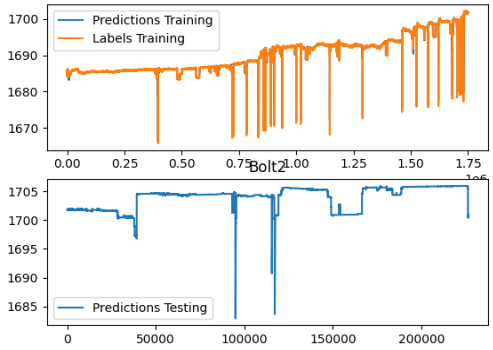

Figure 18: Bolt3 of our final linear constrained boosting model on the actual prediction dataset

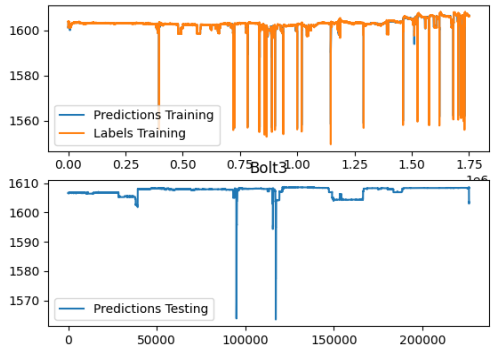

Figure 19: Bolt4 of our final linear constrained boosting model on the actual prediction dataset

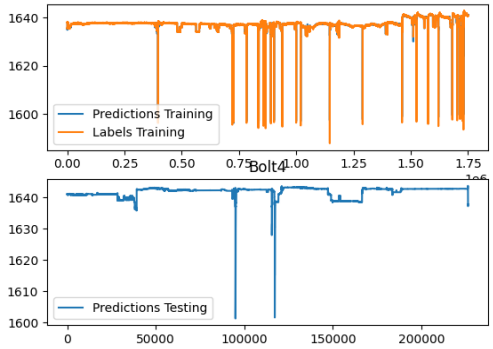

Figure 20: Bolt5 of our final linear constrained boosting model on the actual prediction dataset

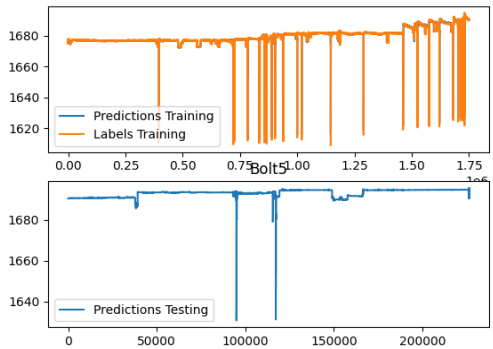

Figure 21: Bolt6 of our final linear constrained boosting model on the actual prediction dataset

This looks descent, but of course it is not possible to say, since we do not know the real labels.

4 Observations/ Insights (focus: interpret the results/ data)

The following insight is gained using the large data set over 6 months. Most of the insight is not applied when training our model, as some of the observed effects in the six month data is not very noticeable in the shorter term, smaller data set. Hence, this insight applies to data over a long term time frame, while when we studied the smaller data set we assumed a local linearity. To look into how the tensions changed over time, we looked at the first data set recorded over six months. We noted that the change in tension on the bolts was unpredictable in some cases. Here, we plot the tension on some of the bolts over time:

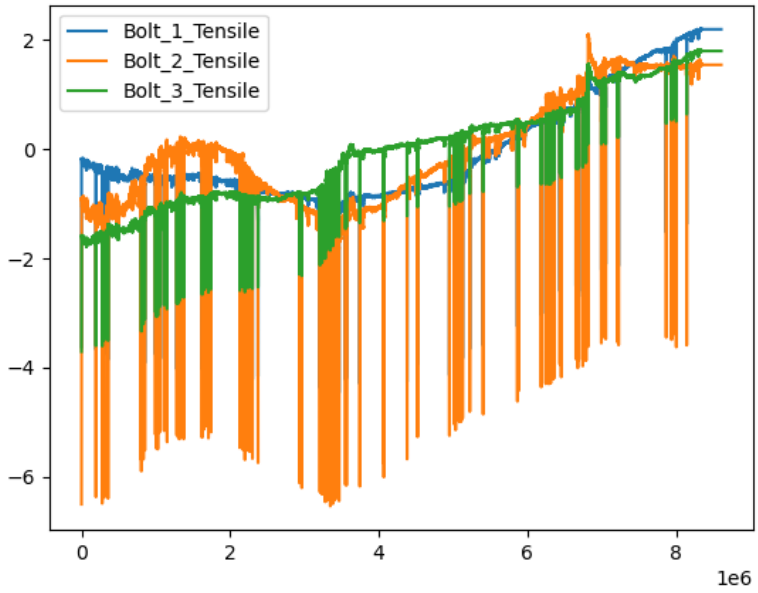

Figure 22: Tensile data of bolt 1, 2 and 3 over 6 months

We then made train/test split on the data in order to fit the model and test it. When attempting to fit a linear regression model, none of the data features could help predict the general trend in the tensile values over time. When we added time as an input feature, the model was somewhat able to predict the trends in tensile. Since the relationship between trend in tensile and time appears non-linear, we also tried adding time squared as a parameter to the model. We plot the results:

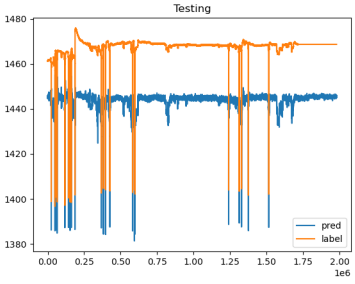

Figure 23: Predicted tensile on bolt 1, only original input features, along with correct values

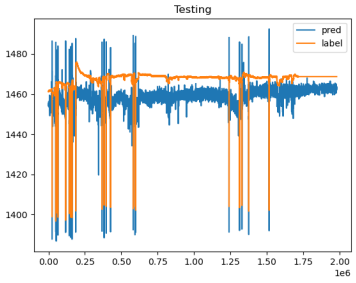

Figure 24: Predicted tensile on bolt 1, using time as input feature, along with correct values

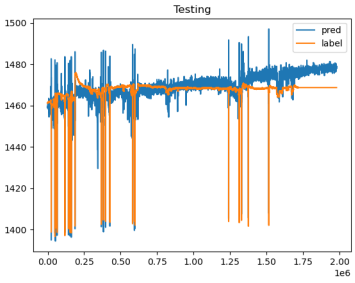

Figure 25: Predicted tensile on bolt 1, using time and time squared as input features, along with correct values

We observe that when adding both time as a feature in the regression helped the model predict the tensile values more accurately, and adding time squared helped predicting even more. This might mean there is some relationship between the time and the tensile values which the data we are given does not contain, and this relationship appears to be non-linear. We note that the only input feature which seems to may have a relationship with the long term trend in the tensile values is the temperature. Here we plot the temperature for bolt 1 along with the tensile for bolt 2:

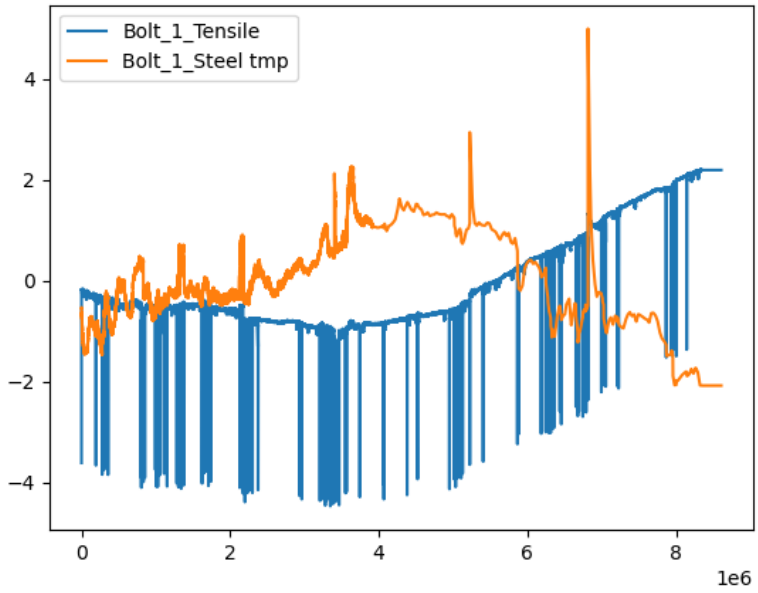

Figure 26: Tensile and temperature of bolt 1

We see that the two values seem somewhat negatively correlated. If this there is a relationship between the tensile and the temperature, the tensile values would presumably change with the seasons. Another theory is that the turbine has "shifted" during the past six months as it was recently modified, and that the tensile will converge to a more stable trend after a while. We note that this correlation is not as high for the tensile of other bolts, so we cannot conclude that this relationship is legitimate, and it remains only a theory. To further explore this correlation we would need more temperature and tensile data, perhaps over a whole year or more.

We conclude that the unpredictable trends in tensile are most apparent in the six month data. Since our main task is to predict the tensile over a period of a few days, we assume that the change is tensile over time is linear in the short term.

5 Github Repo

Github REPO - Hackathon

6 Bibliography

[1] Microsoft Corporation. *Welcome to LightGBM's documentation!* URL: https://lightgbm.readthedocs.io/en/latest/.

[2] scikit learn. *Documentation Linear Regression model.* URL: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html.