Electronics and Computer Science
Faculty of Engineering and Physical Sciences
University of Southampton

Matthew Holmes

July 14, 2023

Machine Learning Modelling for Multivariate Time Series data in Tribology

Supervisor: Jo Grundy

PhD Colleagues: Maruti Sakhamuri and Zaihao Tian

Contents

1 Overview

Within the internship two sets of data were provided for analysis. The test used to produce the data is known as the TE74 Running in Test, and one set of data is produced from the sensors on the machinery and the other is produced from taking surface measurements on the discs before and after the test. For the sensor data the goal is to be able to classify whether pitting has occurred or not, and for the surface measurements the goal is to accurately predict the resulting roughness of one of the discs involved in the test.

2 TE74 Running in Test

2.1 Explanation of the test

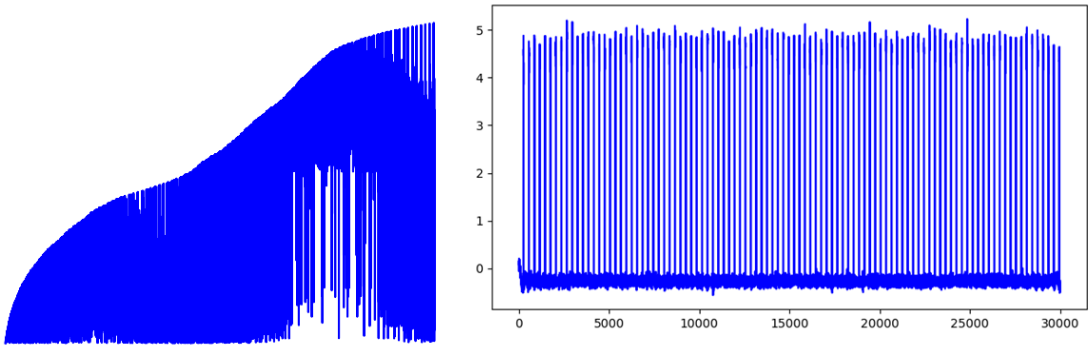
The TE74 test involves using the Twin Roller Tribometer to study traction, wear and rolling contact fatigue of two discs under contact. The tribometer provides high contact pressures and high loads through a servo controlled pneumatic bellows actuator with force transducer feedback. A vibration sensor is used for detecting surface failure, and provides one of the key features that are used to train the model later on. Slip rings are placed on the roller shafts to provide electrical contact resistance measurements for another set of features. The surface measurements represent the roughness of the surfaces, taken both before the test starts and after it has concluded. The two discs in the test are known as 'Cyl' and 'Cro' in the data, and the different measurements of the disks are 'Ra', 'Rq', 'Rp', 'Rv', 'Rsk', 'Rku' and 'Rdq' which correspond to different areas of the disk being measured.

2.2 Data

2.2.1 Sensor Data

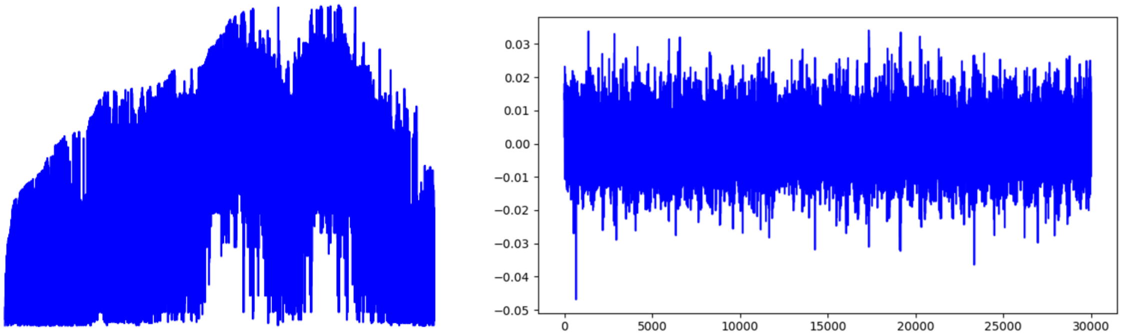
The data produced by the sensors is high dimensional - for each test there are a number of timesteps, each of which are equally spaced apart in real time, and within each timestep there are 4 features (vib, S1, S2 and encoder) which have equal number of samples. The samples for each feature are acquired by retrieving data from the appropriate sensor for 1 second at 3KHz sampling rate. Between each timestep there is no data retrieval done, this only occurs for 1 second at the start of each timestep.

Before working with the Machine Learning (ML) models, the data must first be put through a high pass filter to remove any noise produced during testing. After this, the data can be passed to the ML models. Figs 1 and 2 show the result of filtering the data, showing how much noise there is given the difference between the filtered and unfiltered data.



(a) (b)

Figure 1: (a) Unfiltered 'encoder' (b) Filtered 'encoder'



(a) (b)

Figure 2: (a) Unfiltered 'S2' (b) Filtered 'S2'

Before working on the models, it is important to identify key features within the data that may indicate pitting so it can be determined later if the models are predicting what they should be. Within the feature encoder there are no unusual features, but both 'S2' and 'vib' - which is the vibration data feature - have unusual spikes that are shown in Fig 3. This is believed to be indicative of pitting occurring, especially when this is seen in the vib feature since it means the surfaces are becoming more rough thus producing larger vibrations.

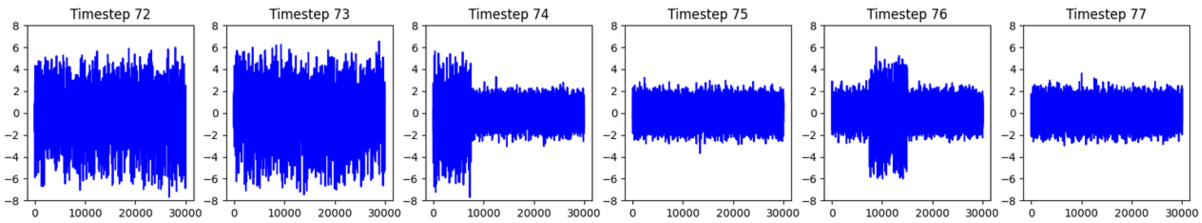


Figure 3: Examples of varying amplitude from feature 'vib'

2.2.2 Roughness Data

The data obtained from making roughness measurements is not as complex as the set of data obtained from the sensors. The data forms a multioutput regression problem, with the input features consisting of initial roughness measurements along with the chosen Pressure, Speed and SRR for the test, and the output features consisting of the post test roughness measurements along with the Lambda and Friction Coefficient, and the number of Cycles To FullFilm (Cycles to FF). The data visualisation for the roughness measurements will be shown in the results where there is a comparison to make between the predicted and actual values.

3 Method

3.1 Sensor Data

After filtering, it was chosen to scale the data to improve learning capabilities of the model. Due to the dimensionality of the data, it was required to have individual scalers for each feature which seems to reduce the usefulness of scaling in the first place, as this means each feature is still on a different scale however they will now be in the same order of magnitude at least. By the end of the work it was not determined if scaling provided any benefit to the results or not.

Since the data is timeseries, an attempt was made to use either LSTMs or GRUs since those are the most manageable types of RNNs. Due to the data being high dimensional it was required to use the ConvLSTM1D from Keras in order to process the data correctly, but this resulted in a limited choice of data processing.

From here, the final two types of model were produced for the data. The first works with the higher dimensions by taking each entire test as a sample, with the whole set of timesteps being used as input to the model at once. The second removes the regard for the timeseries component of the data, and takes each individual timestep as an input to the model. This means that when shuffling the data for training the order of the data will be lost, however this does not matter to this type of model.

3.2 Roughness Data

Since this is a multioutput regression problem, there are limited prebuilt options to work with and there is not enough time to develop custom model implementations that would work with the data. Sklearn has inbuilt multioutput regression support for selected methods through the use of their MultiOutputRegressor object. This object was found to work with Random Forest, Gradient Boost, Linear Regression, SVR, KNeighbours regressor, MLP regressor, Gaussian Process regressor and Decision Trees.

The models perform a fit and predict, so data preprocessing is required to see any further reduction in error. It was determined from the data that some changes could be explored: keeping the standard deviation features in the data, remove the friction coefficient feature, and remove the CyclestoFF feature. The data is also scaled since features such as the speed and CyclestoFF are a few of magnitudes larger than the roughness measurements.

4 Results

4.1 Sensor Data

4.1.1 Effects of scaling

This section looks at the higher dimensional model and the effects on the predicted values that scaling produces.

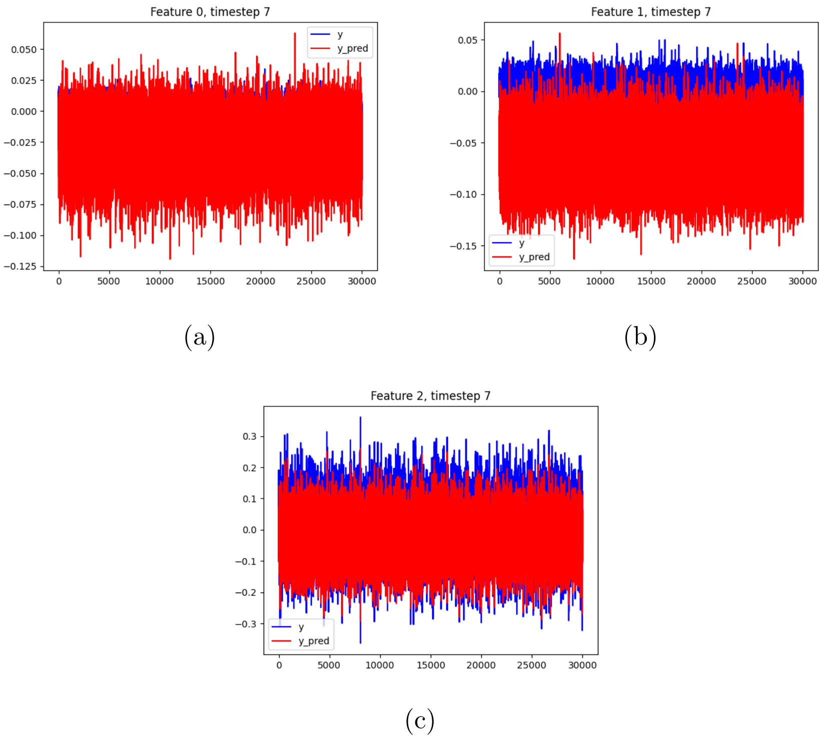
Scale	MSE	MAPE
No	0.00219	33.69
Yes	1.19	2.59

Table 1: Mean Squared Error (MSE) and Mean Absolute Percentage Error (MAPE) for with and without scaling

As can be seen from 1 there is a large difference in the metrics when scaling is introduced. To explain the change in metric values, the MAPE change will be explored using some examples.

Figures 4 and 5 are not scaled and scaled respectively, and the scaling changes not only the range and loss values but also the way the model trains on the data. In the Sklearn documentation for MAPE it is stated that ‘the output can be arbitrarily high when y_true is small (which is specific to the metric) or when abs(y_true - y_pred) is large (which is common for most regression metrics).’

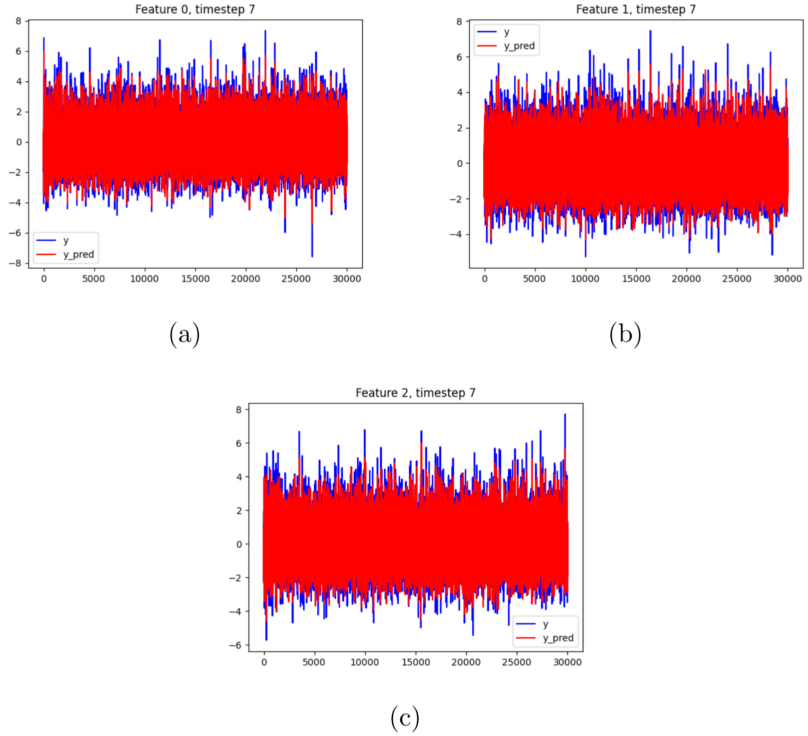
The second part of the statement is stating the basics of an error metric, however the first part provides reasoning for why two of the MAPEs in Fig 4 are so high. The range for the scaled data is [-8, 8] which presumably is not considered a ’small’ value for y_true, however the range of the unscaled data - for features 0 and 1 - is [-0.15, 0.05] which is considerably smaller, and thus resulting in arbitrarily large errors.



(a) (b)

(c)

Figure 4: Unscaled data with MAPEs: (a) 75.6 (b) 333.4 (c) 3.8



(a) (b)

(c)

Figure 5: Scaled data with MAPEs: (a) 2.2 (b) 2.8 (c) 3.3

4.1.2 Looking at prediction trends

This section covers some notable trends that were spotted during the work, the first of which shows that the models had issues with learning the data trends quickly, leading to the predictions of the first and last timesteps to have an unusually small amplitude. This is shown below in Fig 6. but unfortunately the cause of this could not be determined.

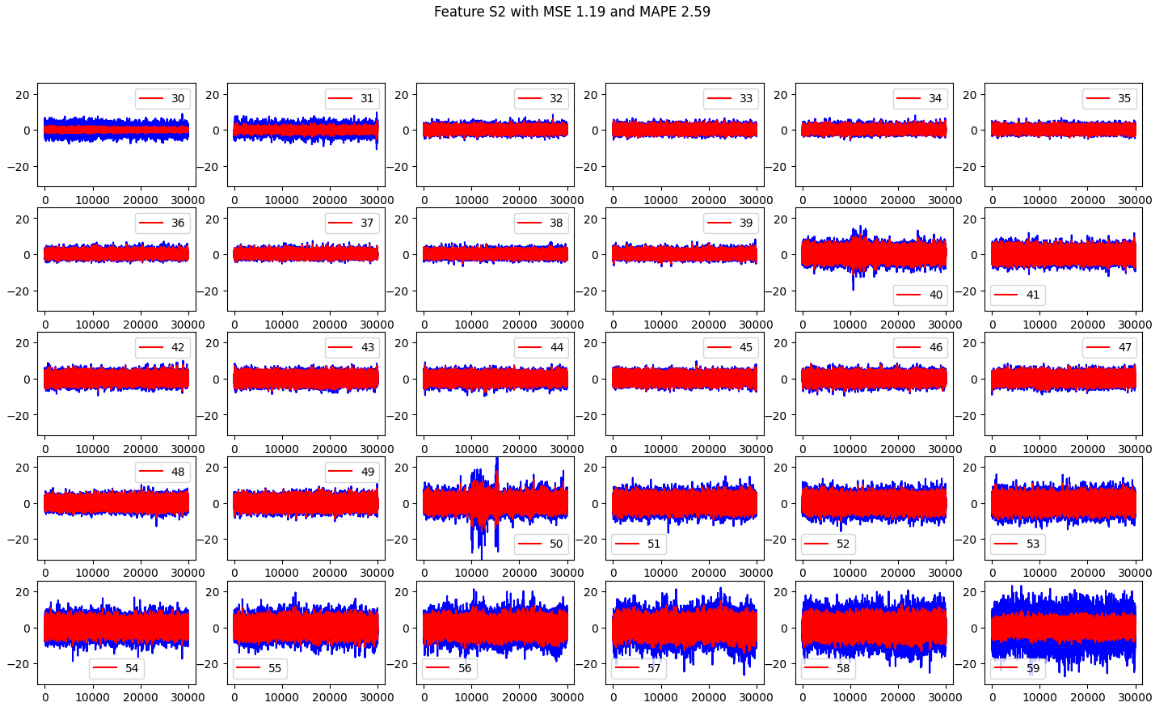


Figure 6: Timesteps 30 to 60 of TE74-base oil-test4

Another type of attempted model was a Recurrent Neural Network (RNN), which aims to take into account previously encountered data for future predictions. Specifically a Long Short Term Memory (LSTM) network was used which an example of the prediction it produced is shown in Fig 7. It can be clearly seen that the model does not learn anywhere near enough to make good predictions, this is most likely due to the fact that the model is not deep enough or that the learning parameters / layers are not set up correctly to allow the model to learn sufficiently. However if the model becomes deeper, it may become apparent that the volume of training data is not sufficient to train a larger model.

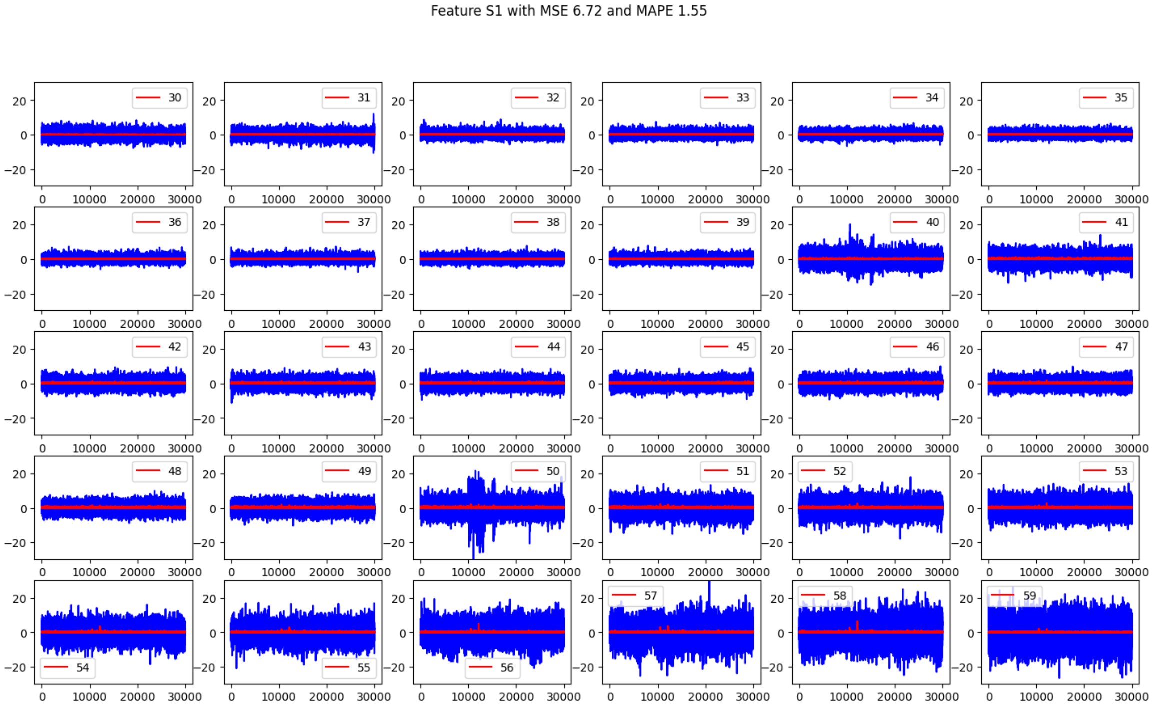


Figure 7: Timesteps 30 to 60 of TE74-base oil-test4

4.1.3 Lower dimensional model

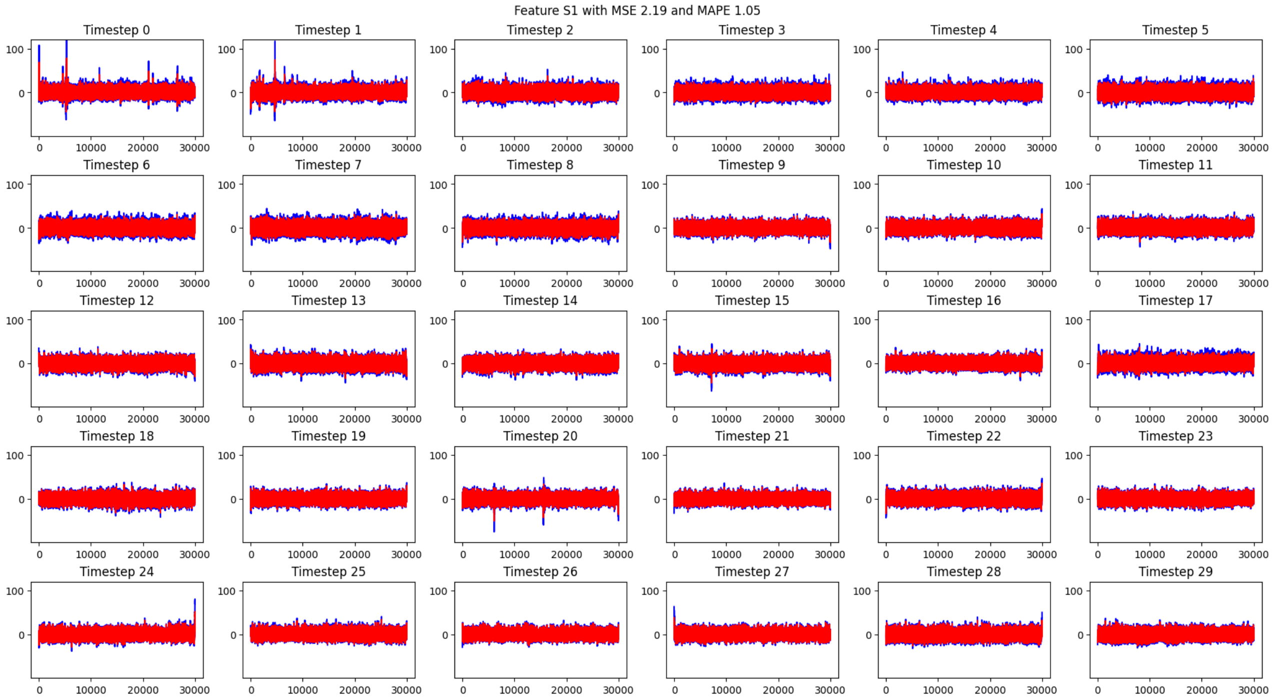


Figure 8: Timesteps 30 to 60 of TE74-base oil-test4

By comparing Fig 6 and Fig 8 it can be seen that the lower dimensional model does not suffer from lag at the beginning of the prediction since for this model, each timestep is independent and thus there is no considered 'start' and 'end' of the test. The model still succeeds to predict spikes and changes in amplitude as can be seen in Fig 9.

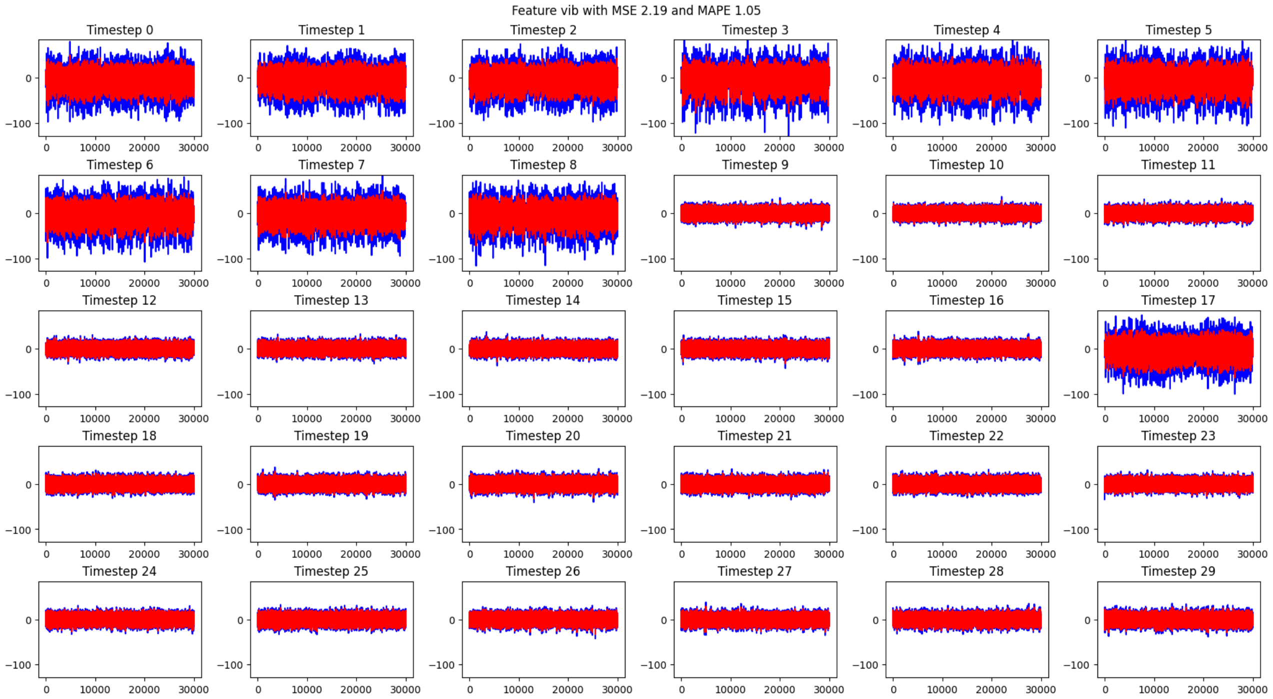


Figure 9: Timesteps 30 to 60 of TE74-base oil-test4

4.1.4 Final comparisons

Model	MSE	MAPE
Normal CNN	**1.19**	2.59
Lower dimensional	2.19	**1.05**
RNN	6.72	1.55

Table 2: Mean Squared Error (MSE) and Mean Absolute Percentage Error (MAPE) for with and without scaling

The results are not fully comparable since the lower dimensional model was tested against all 96 timesteps, but the normal model and RNN are only able to test against exactly 30 timesteps at a time. The RNN is by far the worst performing model, although the MAPE score is misleading in its case as confirmed by Fig 7. it can be seen that the prediction is entirely incorrect.

From this table and previously discussed results it has been determined that the lower dimensional CNN is the best model to use for this data, since its predictions look the most accurate and it is able to handle varying numbers of timesteps making data collection easier.

4.2 Roughness Data

4.2.1 Effects of scaling

For this section the prediction Mean Squared Error (MSE) was calculated for every ML model with and without scaling, shown in Table 3. The scaling function is the StandardScaler implemented by Sklearn, and utilises one call of the fit_transform() function for the whole dataset before then splitting the data into input (X) and output (y) data.

	MSE	
Model Name	With scaling	Without scaling
Random Forest	0.80	2196319.58
Gradient Boost	1.16	3674437.52
SVR	0.59	309392.39
KNeighbours	0.77	5780449.58
MLP	1.80	644263.95
Gaussian Process	0.63	4470525.91
Decision Tree	2.06	1387603.05
Average Loss	**0.98**	2307873.44

Table 3: Mean Squared Error for models with and without scaling

The difference in the results can be explained by looking at the individual output feature losses, shown in Table 4. Only the first 6 features were chosen to view, as the remainder contain losses around the 10^{-5} range. As it can be seen from the individual features Friction Coefficient and CyclesToFF, those features have extremely large losses in comparison to every other feature when there is no scaling, however when there is scaling those features have similar losses to other features. This may be that the models perform better when scaling is introduced, but also could be an effect of reducing the range of values which would naturally decrease the loss. It should also be noted that the loss of the standard deviation features increase more than other features after scaling.

Individual Feature Loss for Random Forest Regressor		
	MSE	
Feature Name	With scaling	Without scaling
CompRqPost	0.081	1.92e^{-6}
LambdaPost	0.085	0.0201
Friction Coefficient	0.450	192.96
CyclesToFF	0.424	6.15e^{7}
RaCylPost	0.588	6.25e^{-6}
RaCylPost_Std	1.207	3.30e^{-6}
Average Loss	**0.47**	10250032.42

Table 4: Individual feature loss for a chosen ML model

In the next sections, the effects of the CyclesToFF feature, Friction Coefficient feature and the standard deviation features will be explored. For those sections, scaling will be implemented in order to obtain results which are comparable between features to properly measure their impact.

4.2.2 Effects of the CyclesToFF feature

	MSE	
Model Name	With CyclesToFF	Without CyclesToFF
Random Forest	0.80	0.82
Gradient Boost	1.16	1.18
Linear Regression	15.03	15.31
SVR	0.59	0.59
KNeighbours	0.77	0.76
MLP	1.80	1.82
Gaussian Process	0.63	0.63
Decision Tree	2.06	2.12
Average Loss	**0.98**	0.99

Table 5: Mean Squared Error for models with and without CyclesToFF feature

As it can be seen from Table 5 there is no major difference between including or excluding the CyclesToFF feature. A closer look at the predicted data in Table 6 shows that the Random Forest model mostly just predicts the same three values: -1.0, 0.8 and 3.2. The scaler also provides a false interpretation of the data by making some values negative when they shouldn't be, since a -1.0 value for this feature signifies full film was not achieved. A potential solution for this could be to try a classification algorithm first to filter out whether full film was achieved, followed by a regression solution to find the number of cycles to full film along with predicting the rest of the output features.

Sample index	Predicted	Actual
0	0.8	-0.2
1	-1.0	-1.0
2	-1.0	-1.0
3	-0.4	-0.7
4	0.8	-1.0
5	-1.0	0.7
6	3.2	0.4
7	-1.0	-0.2
8	3.2	-0.7
9	-0.4	-1.0
10	0.8	1.0
11	-1.0	1.3
12	0.8	-1.0
13	3.2	0.2
14	-0.6	1.9

Table 6: Comparison between the Actual and Predicted values of CyclesToFF feature produced by the Random Forest model

4.2.3 Effects of the Friction Coefficient feature

	MSE	
Model Name	With Friction Coefficient	Without Friction Coefficient
Random Forest	0.80	0.81
Gradient Boost	1.16	1.18
Linear Regression	15.03	15.36
SVR	0.59	0.58
KNeighbours	0.77	0.77
MLP	1.80	1.78
Gaussian Process	0.63	0.60
Decision Tree	2.06	2.11
Average Loss	**0.98**	**0.98**

Table 7: Mean Squared Error for models with and without Friction Coefficient feature

Again Table 7 shows that there is little difference in including or excluding the Friction Coefficient feature, so it will be left in for the time being.

4.2.4 Effects of Standard Deviation (Std) features

	MSE			
Model Name	With Std	Without any Std	Without Std input	Without Std output
Random Forest	0.80	0.62	1.01	0.74
Gradient Boost	1.16	0.92	1.28	1.15
SVR	0.59	0.51	0.78	0.56
KNeighbours	0.77	0.76	0.98	0.87
MLP	1.80	0.60	1.16	1.74
Gaussian Process	0.63	0.66	0.66	0.68
Decision Tree	2.06	1.85	2.07	2.05
Average Loss	0.98	**0.74**	1.0	0.97

Table 8: Mean Squared Error for models with and without Standard Deviation features

From Table 8 it can be seen that the best result is obtained by excluding all of the Std features from the data. This is most likely due to the fact that the standard deviations are all very similar values and do not provide an useful information or patterns for the models to utilise, in fact the consistency of the values probably reduces the learning of other features for the model.

4.2.5 MAPE and MSE graphs

In this section a visualisation of all the output results is produced to provide a full comparison of the roughness data, which can be seen in Figures 10 and 11. Focusing on the MSE results in Fig 11 it can be seen that for the first four features there are many models which achieve an MSE of lower than 0.25, however for the remaining roughness features only Rdq1Post gets consistent losses lower than 0.5. Upon further inspection of the data, this could be to do with the difference in the magnitude of the data - since all other features apart from Rku1Post have values in the 10^{-2} range compared to 10^{1} - however this would be dealt with by the scaler. As such it is unknown why there is variation in the roughness feature losses.

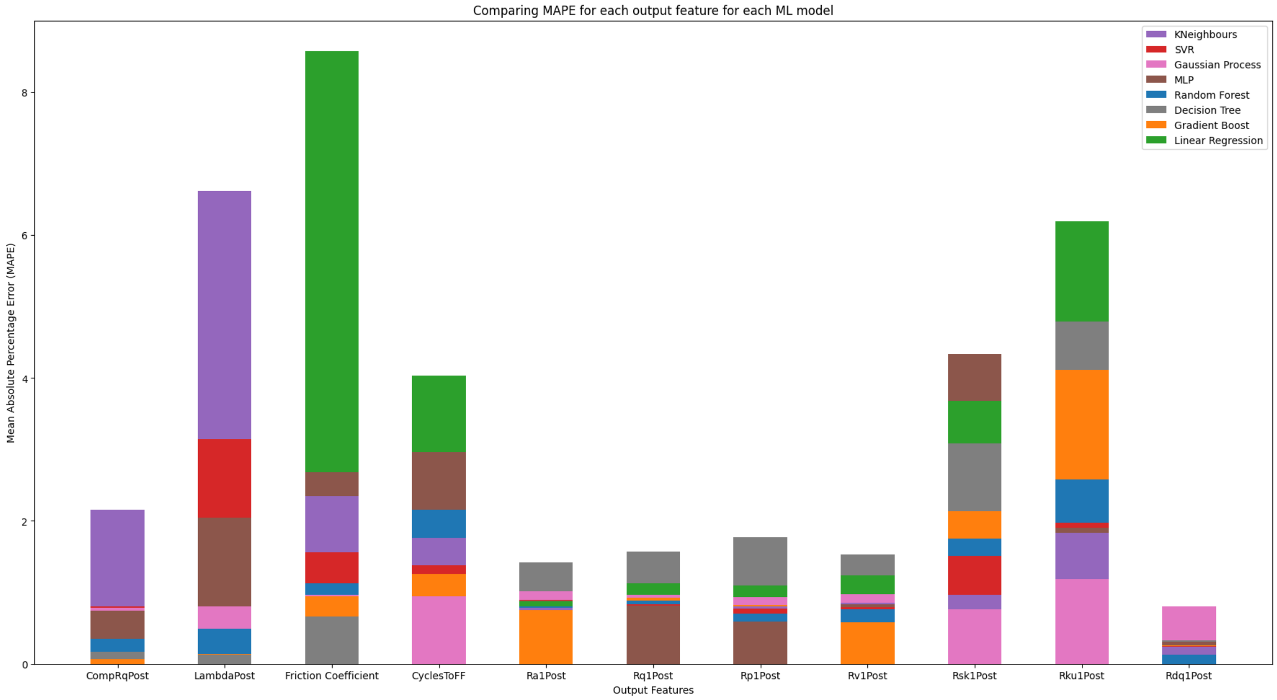


Figure 10: MAPE for each output feature for each ML model

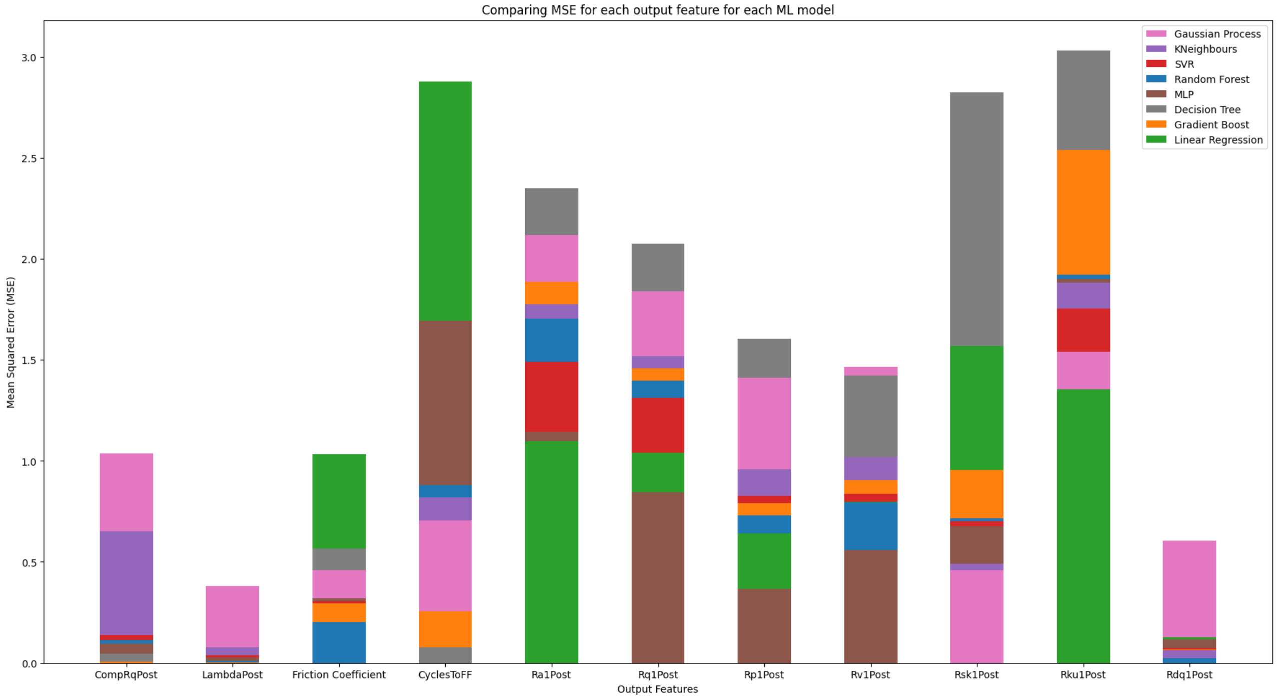


Figure 11: MSE for each output feature for each ML model