UNIVERSITY
OF GÄVLE

FACULTY OF ENGINEERING AND SUSTAINABLE DEVELOPMENT
Department of Electrical Engineering, Mathematics and Science

Data Analysis for Predictive Maintenance of a Straightening Machine in the Steel Industry

Paul Barron

September 2023

Student thesis, Advanced level (Master degree, one year), 15 HE
Electronics
Master Programme in Electronics/Automation (online)

Supervisor: Daniel Rönnow
Assistant supervisor: Oscar Bautista Gonzalez
Examiner: José Chilo

Acknowledgments

I would like to express deep gratitude to my supervisors, Daniel Rönnow and Oscar Bautista Gonzalez, for their time, guidance and support. To the University of Gävle I am extremely appreciative for opportunity to complete a Masters Program in Electronics and Automation available as distance education. This is ideal for full time working professionals and is the only course of it's kind in Sweden. Special thanks to Alleima for providing the data to analyze for this Thesis, without which it would not have been possible. Finally, I would also like to acknowledge the fact that I am undertaking this study for no personal cost thanks to free education offered by the Swedish Government. More countries should consider offering free education. Knowledge is power.

Abstract

The availability of industrial machinery is crucial to any business operating in the manufacturing sector. Mechanical failures halt production and unplanned downtime can be disruptive and costly. Small failures can compound to serious failures which exponentially increases downtime and repair costs. Therefore Identifying a degradation condition before reaching failure is key to maintaining machine availability. On the other hand, it's undesirable to spend resources performing maintenance that is not required. For these reasons a large field of academic work is dedicated to analyzing the health of a machine, it's remaining life and in turn preventing failures.

This thesis analyses data from a tube straightening machine used in the steel industry with the goal of implementing a condition monitoring strategy. The data comes from a real world application provided by a multinational manufacturer of steel products. It was obtained using the existing sensors and data acquisition system. The project serves as a study of the existing infrastructure (available sensors) and it's suitability for implementing a condition monitoring strategy. The work is the first step in a larger study and does not attempt to perform any implementation or fault identification. In a broad sense the aim of the project is to identify relationships and patterns in the data that could be varying with time as the machine degrades.

The data consists of twelve channels taken over a two week duration. It is prepossessed to isolate periods where the machine is operating and separated into cycles. Each of these is then further processed to extract time and frequency domain features. The features within each channel are compared with each other using the R_2 coefficient of determination to find combinations that are correlated. A semi automated process is used to select the feature combinations. The same process is performed between signals for each feature.

A number of linear regression models are created based on the results from the correlated features as well as some multivariate models. These are then compared using a goodness of fit metric, Normalized Root Mean Square Error (NRMSE). Potential clustering of machine states are highlighted based on observations in the feature combinations. The conclusions drawn from this study include identification of correlations between signals, potential non-linear relationships and suggestions for future data collection and analysis going forward. No one feature was identified as correlated between all signals.

Contents

List of Figures

List of Tables

Acronyms

CM Condition Monitoring.

FFT Fast Fourier Transform.

NRMSE Normalized Root Mean Square Error.

PCA Principal Component Analysis.

PM Predictive Maintenance.

RMS Root Mean Square.

SINAD Signal-to-noise and Distortion Ratio.

SNR Signal to Noise Ratio.

SSR Sum of Squared Regression.

THD Total Harmonic Distortion.

TSS Total Sum of Squares.

1 Introduction

1.1 Background

Condition Monitoring (CM) is the process of predicting machine health through a combination of sensor data and software. Sensors are used to measure parameters (vibration, temperature, acoustic emissions etc.) which provide inputs to an algorithm that outputs health metrics. This allows operators to gain an understanding of when a machine is likely to fail and instead plan and perform preventative interventions. The ability to monitor equipment health and predict remaining lifetime has valuable benefits. Avoiding unplanned downtime allows the extraction of more value from existing resources and less disruption to production schedules.

The application of CM and Predictive Maintenance (PM) is a large area of research in the machine and manufacturing sector. Of course it is not only limited to machines, being utilized on other engineering structures like bridges [1], process plants like in the oil and gas industry [2] and also biological systems [3]. A common application of CM and PM in manufacturing is the vibration analysis of rotating machines [4], [5]. Breakdowns in these types of machines are most commonly caused by failures in bearing subsystems. The degradation of a bearing results changes in the vibration signal, among others, which can be a indicator of bearing health and remaining life [6].

One such example of an industrial machine that uses roller bearings is a rotary tube straightening machine [7]–[13]. These are used as a finishing process in the steel industry for the manufacturing of metal tubes. They are crucial for straightening the tube in the longitudinal direction and improving ovality of the product. Straightening machines, like many other industrial machines, are expensive to procure and down time is costly. Performance may also be impacted by the machine being in poor condition and in need of maintenance. Thus it is in the interest of operators to be able to schedule planned maintenance instead of having unplanned failures and downtime.

While there is a vast array of research around bearing vibration and CM, all machines have their nuances so one CM strategy is not necessarily suitable for all. No research specifically involving CM on a straightening machine could be found. Unlike a simulation study, CM requires real-world data which is not always readily available unless employed by the company owing the machine. This study assesses real world data from such a machine at a Swedish steel manufacturer, Alleima.

One of the challenges in implementing CM is that there are often huge amounts of data to process and sift through. Feature extraction can be used to reduce the amount of the data whilst preserving the information. By analyzing the features one can then select the most relevant channels and reduce the dimensionality of the data by removing irrelevant, highly correlation or noisy signals. There exist many automated and non-automated methods to select the best features and is a current area of research. This study uses a semi-automated method using the R_2 correlation coefficient to select the most appropriate features.

CM often uses supervised learning which utilizes labeled data to perform classification, i.e. data is labeled as normal or fault condition. Another difficulty with CM research is the limited access to failure data meaning one is often forced to implement unsupervised learning methods. It is common to have partially labeled or totally unlabeled data and thus a different set of techniques exist for unsupervised learning. Here the data analysis is geared towards pattern recognition and clustering since no failure information is available to go with the data. This study has no access to fault or maintenance data and therefore has the goal of trying to find patterns in the data, identify potential clustering opportunities and look for trends over time.

1.2 Project Aim

The aim of this project is to examine the existing data available for a tube straightening machine and make some observations as to whether it is possible to perform CM and PM for this machine. The first step is to analyze the available data that can be be attained using the existing sensors and data acquisition system. The end goal is to begin developing a strategy that will assist the company to know when the machine requires maintenance. The specific goals of this project are summarized as follows:

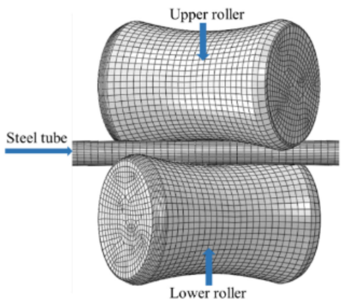
- Understand what signals are related to each other.
- Understand the signals from a statistical point of view.
- Calculate signal features, identify which are related to each other and whether or not they are useful for condition monitoring i.e. which features may represent degradation conditions.
- Combine appropriate features to create data models and identify trends.

2 System

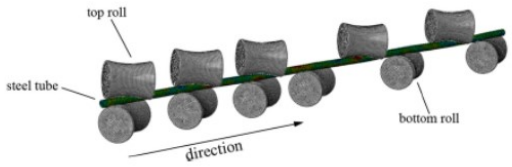
Rotary straightening machines, figs. 1 and 2, are a type of finishing machine developed to straighten and improve ovality of a metal pipes and tubes. A tube is fed through a series of roller pairs and undergoes a sequence of bending moments which induce controlled plastic deformations. Tubes are subjected to two types of straightening forces, pressure straightening and bend (or offset) straightening. Hydraulic actuators are used to precisely position the height of the rollers and at least some of the rollers are driven by motors which grip and rotate the tube while feeding it through the machine. A number of different configurations for this type of machine exist with six and 10 rolls being common.

The profile of the rollers is not the same as the tube radius but are hyperbolic in shape and only contact the tube in 3 places. This shape allows it to accommodate different tube diameters by adjusting the gap or height between the upper and lower rollers as well as the angle between them and the tube. The rollers can be adjusted from parallel to the pipe up to approximately 45 degrees, with the upper rollers being adjusted in the opposite angle to lower rollers.

Modern straightening machines are equipped with sensors that monitor the amount of force being applied, the deformation of the tube, and other relevant parameters [14]. This feedback allows the machine to make real-time adjustments to the straightening process for optimal results. These sensors, while controlling the process itself can also be used for condition monitoring purposes. Significant research exists on the mechanical analysis of such machines [7]–[13] however no studies relating to condition monitoring of such systems using real life data could be found.



(a) Straightening machine roller profile [9]



(b) 10 roller straightening machine layout [13]

Figure 1. Diagrams of straightening machines

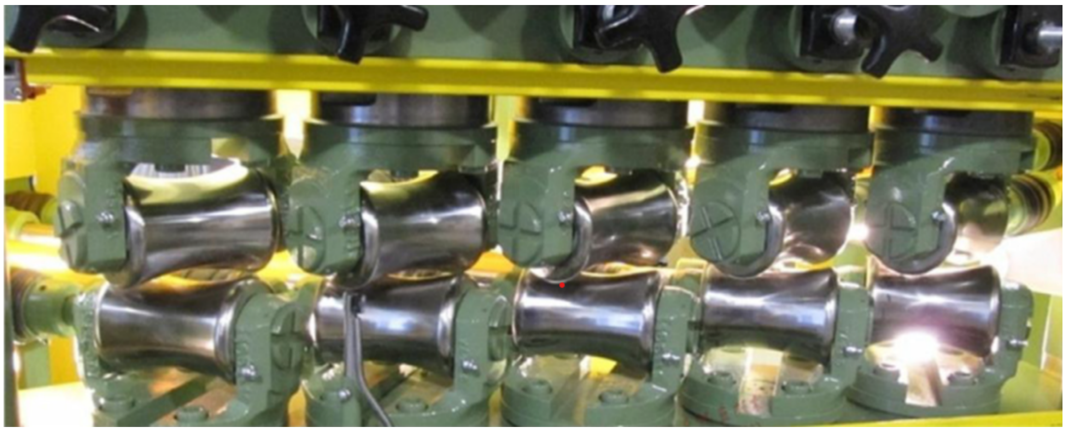


Figure 2. Image of a 10 roller straightening machine [13]

3 Theory

3.1 Feature Extraction

Machine learning algorithms suffer from high amounts of data and information redundancy while requiring vast amounts of storage. Feature extraction, the process of computing numerical values from raw signals, is one method that can be used to combat these issues. By extracting signal features and deciding which are irrelevant one can reduce the amount of data that needs to be stored and processed while still retaining the essence of the signals.

Feature extraction techniques are commonly used to estimate degradation trends in machines [15]–[17]. Features can include time domain, frequency domain, time-frequency domain (temporal) and model based methods [18]. Some of the most widely used and effective time domain indicators are mainly based on Root Mean Square (RMS), crest factor, and kurtosis [19]. Time-frequency values are three dimensional features which include wavelet analysis, spectrograms, scalograms, Wigner-Ville distribution, and kurtograms [19]. [20] proposes a feature set built on experimental data from sensors plus results from building a simplified kinematic and dynamic model.

Machine CM can utilize various form of sensors and signals including: vibration monitoring, acoustic emission monitoring, a fusion of vibration and acoustic, electric motor current monitoring, oil analysis, thermography [21]. There is no one size fits all when it comes to condition monitoring. For a slewing bearing, which is a large low-speed heavy-load bearing, vibration signals are insufficient [22] and thus fusion models based on torque, temperature and vibration are needed.

3.2 Signal Features

A non exhaustive list of features, in particular the ones used in this project, are as follows [23]:

The mean, $\bar{x}$, is the average value of the signal,

$$\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N}, \tag{1}$$

where x_i is the i:th element of the signal and N is the number of samples in the signal.

The standard deviation, σ, is the square root of variance, and measures the dispersion of a data set relative to its mean,

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{(N-1)}}. \tag{2}$$

RMS is the square root of the mean squared,

$$x_{RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}. \tag{3}$$

Excessive variation in the RMS level usually means a change in health status and possible failure.

Shape factor is the root mean square divided by the mean of the absolute value,

$$x_{SF} = \frac{x_{RMS}}{\frac{1}{N}\sum_{i=1}^{N}|x_i|}. \tag{4}$$

It is dependent on the signal shape and independent of the scale of the signal.

Kurtosis is derived from the statistical moment of the fourth order and is defined as the ratio between the mean value of the signal raised to the power of 4 and the square of its variance,

$$x_{kurt} = \frac{\sum_{i=1}^{N}(x_i - \bar{x})^4}{(N-1)\sigma^4}. \tag{5}$$

It is used to analyze the flattening of a distribution and observe the shape of the signal. It is a statistical measure of the tailedness of a distribution and indicates the total degree of outliers present. An increase in the number of outliers, and thus an increase in the Kurtosis, can be a indication of an impending fault. For normal bearing operation, kurtosis is close to 3 and increases rapidly with failure.

Skewness is a measure of asymmetry in the probability density function,

$$x_{skew} = \frac{\sum_{i=1}^{2}(x_i - m)^3}{(N-1)\sigma^3}. \tag{6}$$

Faults can sometimes be revealed through asymmetric distribution.

The maximum value of a signal,

$$x_p = \max(x_i), \tag{7}$$

where x_p is the peak value of a signal.

The maximum absolute value divided by the mean of absolute value,

$$x_{IF} = \frac{x_p}{\frac{1}{N}\sum_{i=1}^{N}|x_i|}, \tag{8}$$

in other words a comparison of the height of the peak to the average value.

Crest Factor is found by dividing the maximum absolute value of a signal by the RMS value,

$$x_{crest} = \frac{x_p}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}x_i^2}}. \tag{9}$$

Faults can often be detected as a peakiness of signals before they become evident in the energy (RMS). Other indicators have been developed based on the crest factor, such as the K-factor, by multiplying the peak value by the rms value or the peak-to-peak value, measuring the difference between the amplitudes of the upper and lower peaks [19].

Clearance factor is the defined as the peak value divided by the squared mean value of the square roots of the absolute values,

$$x_{clear} = \frac{x_p}{(\frac{1}{N}\sum_{i=1}^{N}\sqrt{|x_i|})^2}. \tag{10}$$

For a healthy bearings this feature is maximum and decreases as faults develop.

The next features come from the frequency domain and one must first calculate the Fast Fourier Transform (FFT) before computing the features themselves. Signal to Noise Ratio (SNR) is the ratio of signal power to noise power in decibels,

$$SNR = 10\log_{10}\frac{P_{signal}}{P_{noise}}, \tag{11}$$

where P_{signal} is the average power of the signal and P_{noise} is the average power of the noise.

Signal-to-noise and Distortion Ratio (SINAD) is the ratio of total signal power to total noise-plus-distortion power,

$$SNR = 10\log_{10}\frac{P_{signal} + P_{noise} + P_{distortion}}{P_{noise} + P_{distortion}}, \tag{12}$$

where P_{signal} is the average power of the signal, P_{noise} is the average power of the noise and $P_{distortion}$ the average power of the distortion.

Total Harmonic Distortion (THD) is calculated as the ratio of total harmonic power to fundamental power,

$$THD = \frac{P_h}{P_f} = \frac{\sqrt{P_{h2}^2 + \ldots + P_{hN}^2}}{P_1}, \tag{13}$$

where P_f is the power of the fundamental frequency and P_h is the power of the the remaining harmonics. P_N is the N:th harmonic of the FFT and MATLAB takes the first five harmonics using a modified periodogram of the same length as the input signal. The modified periodogram uses a Kaiser window.

3.3 Spectral Features

In order to compute spectral features one must first compute the power spectral density which indicates the distribution of power per unit frequency. MATLAB does this via periodogram method. It then uses the command, 'findpeaks', to locate the maximum values of the power spectrum. A local peak is classified as a sample which is either larger than the two neighboring samples or is equal to infinity. Two features come directly from the peak data 1) Peak Frequency is the value of the frequency which has the highest peak and 2) Peak Amplitude is the height of this same peak on the power spectrum. The Band Power is the area beneath the power spectral density graph, within a selected frequency range. MATLAB uses the command, 'trapz', to compute the approximate integral of the signal via the trapezoidal method with unit spacing.

3.4 R^2 correlation

The coefficient of determination, or R^2 correlation, is a measure of how well a model predicts the actual output values. It is calculated as the one minus the Sum of Squared Regression (SSR) divided by Total Sum of Squares (TSS),

$$R^2 = 1 - \frac{\text{SSR}}{\text{TSS}} = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}, \tag{14}$$

where y_i is the i:th output, $\hat{y}_i$ is the predicted value of the model and $\bar{y}$ is the mean of y [24]. An R^2 value of 1 means that the two signals are perfectly identical expect for a multiplicative constant and a value of zero means there is no linear relationship. A value between 0 and 1 indicates some degree of linear relationship exists with the degree of linearity increasing with the value.

3.5 Feature Selection

Once features have been calculated the challenge is to isolate the most useful and relevant features. Often data sets contain irrelevant, highly correlated or noisy features that can be removed without a significant loss of information. Ideally non-informative and redundant features are removed to reduce complexity, making models easier to interpret. To this point, [3] argues that several widely used classification algorithms can generate misleading feature rankings when the training datasets contain large groups of correlated features. Considering all the research, there is no consensus on which feature/combination of features are best suited for the identification of and distinguishing between faults.

A common method for selecting the most suitable features is forward or backward step wise selection [24]. Forward selection begins with a null model and creates simple linear regressions for each feature on it's own and the model with the lowest chosen metric is selected. Using that model as a starting point for the next stage another variable is added to create a number of 2 variable models with the lowest metric being selected to more forward. This process is continued until some stopping rule is satisfied.

Principal Component Analysis (PCA) appears to be another popular technique used in feature selection. In one study, PCA is used to fuse multiple features in order to obtain a bearings state [25]. Another study uses a PCA-based approach to selecting the most representative features for the classification of defective components and defect severity in three types of rolling bearings [26].

Among other techniques [1] uses recursive feature elimination and feature correlation clustering, the latter involves using thresholds and removing highly correlation features. Usually some kind of goodness of fit metric is used to compare feature models, Zang proposes a combination of correlation, monotonicity and robustness for feature selection [6].

3.6 Linear Regression Models

Once a subset of features has been selected linear models can be created between two features,

$$\mathbf{Y} = \boldsymbol{\theta}\mathbf{X}, \tag{15}$$

where $\mathbf{X}$ is the input vector (or the feature 1), $\mathbf{Y}$ is the output vector of the model (or feature 2) and $\boldsymbol{\theta}$ are the model values. By plotting one signal feature against another signal feature, linear relationships in the data can be revealed. Not only linear relations but quite possibly non-linear relationships too. The parameters, $\boldsymbol{\theta}$, of the linear model are estimated using linear least squares,

$$\hat{\boldsymbol{\theta}} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{Y}, \tag{16}$$

where $\hat{\boldsymbol{\theta}} = \begin{bmatrix} \hat{\theta}_1 \\ \vdots \\ \hat{\theta}_n \end{bmatrix}$, $\mathbf{X} = \begin{bmatrix} x_1(1) & \dots & x_k(1) \\ \vdots & \ddots & \vdots \\ x_1(n) & \dots & x_k(n) \end{bmatrix}$ and $\mathbf{Y} = \begin{bmatrix} y(1) \\ \vdots \\ y(2) \end{bmatrix}$ [24].

Equation 16 is used for multivariate models as well as two variable models. Once a model has been calculated a normalized goodness of fit function [27], NRMSE, can be used to assess the accuracy of the model,

$$NRMSE = 100 \times \frac{1 - \|\mathbf{Y} - \hat{\mathbf{Y}}\|}{\|\mathbf{Y} - \bar{\mathbf{Y}}\|}, \tag{17}$$

where $\mathbf{Y}$ is an $(N \times 1)$ vector containing the the original feature signal selected as the model output, $\hat{\mathbf{Y}}$ is an $(N \times 1)$ vector containing the model's output, $\bar{\mathbf{Y}}$ is an $(N \times 1)$ vector containing the mean value of $\mathbf{Y}$ and ||.|| denotes the Euclidean norm.

3.7 Classification and Clustering

In a perfect world, data would be available for every kind of failure mode possible on a machine. If this were true, supervised learning could be used to train an algorithm to recognize all kinds of degradation conditions and faults. One could build a model that predicts or estimates degradation metrics quite easily based on one or more inputs. This is often not the case and a more likely situation is where a number inputs or parameters exists but no quantitative outputs. However, one can still learn about the relationships between signals and the structure from such data using unsupervised learning. The aim here is to identify patterns in the input-output signals from unlabeled data. Thus, it is useful for modeling complex data for which prior knowledge is hard to get [28].

Unsupervised learning is more of an inference task rather than prediction. Techniques such as auto-encoders, K-means clustering and sparse coding are used to find clusters and patters. Once feature models have been created, one can often find different clustering of data when a machine is in a degraded condition compared to a healthy state.

Many studies are performed in the lab on specially constructed test rigs which are built to run until destruction [19], [20], [22], [26]. This is not possible on an operational asset since a company cannot afford to run the machine until it fails. For this reason studies into CM of real life machines are often forced to implement unsupervised learning as opposed to supervised.

4 Data Processing

4.1 Description of Data

The data provided by Alleima consists of approximately a one hour period of data each day over a 15 day duration. It is assumed that it was taken from the same period of time each day. The data was captured by an Iba-DAQ data acquisition system with a sampling time 0.02 seconds [29]. The list of channels provided are shown in Table I.

Table I. Signal Names

Signal Index	Signal Number	Signal Name
1	21:07	Angle over rolls [deg]
2	21:10	Position over rolls [mm]
3	21:12	Actual moment over rolls [Nm]
4	21:17	Angle under rolls [deg]
5	21:20	Actual moment under rolls [Nm]
6	21:28	Vibration measurements [mm/s]
7	21:31	Width position [mm]
8	21:32	Height position [mm]
9	21:33	Error position for height [mm]
10	21:34	Error position for the width [mm]
11	21:35	Set point force [kN]
12	21:36	Actual force [kN]

4.2 Pre-processing

State detection for this application is performed using the Signal 5, 21:20 Actual moment under rolls. This was suggested by the team at Alleima, who have expert knowledge with the system, as the most appropriate signal to use. When the machine is not actively processing a tube this value is much less than when it is processing a tube. When no tube is in the system there is none or minimal torque on this sensor and when processing it experiences a much larger moment. The data is separated into cycles using this signal to isolate the periods where the machine is in the act of processing a part. A threshold of 500 samples, 10 seconds, was used to filter out short pulses which aren't believed to be full cycles.

Figure 3 shows the windows that were identified from each file where the 21.20 signal was above the threshold. Two files, B_30_03 and B_04_04, contained no identified pulses. In total 240 pulses were identified where the 21.20 signal crosses the threshold. Any pulse under 500 samples was removed from the data for the next stage leaving 224 pulses to extract features from. This was done to filter out transitions where the value jumps briefly above the threshold then back below it. A value of 500 was chosen based on intuition since there was a large gap in between the shortest pulse deemed to be legitimate, 20 seconds, and the longest pulse deemed to not be legitimate, less than 10 seconds.

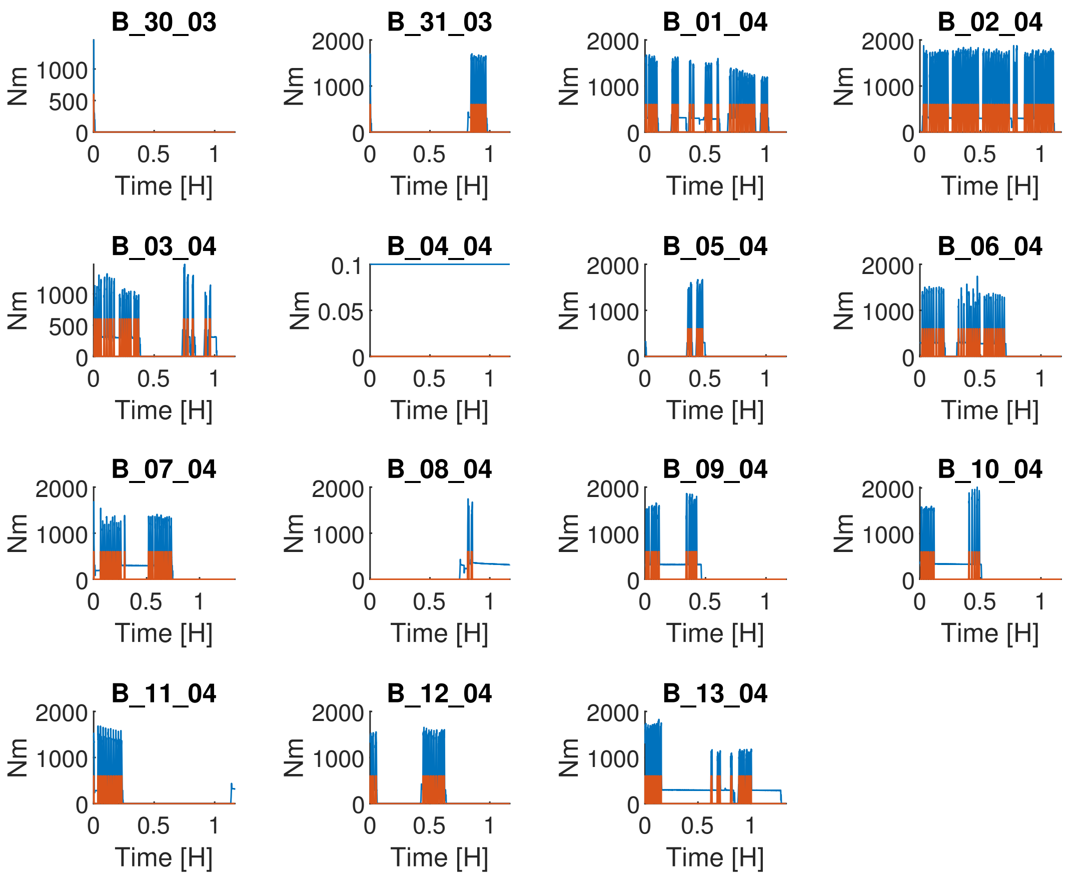


Figure 3. Signal 5 21:20 Actual moment under rolls and State Detection

Figure 4 shows one of the plots from fig. 3, B_02_04, in greater detail. This particular file is the one with the highest number of pulses.

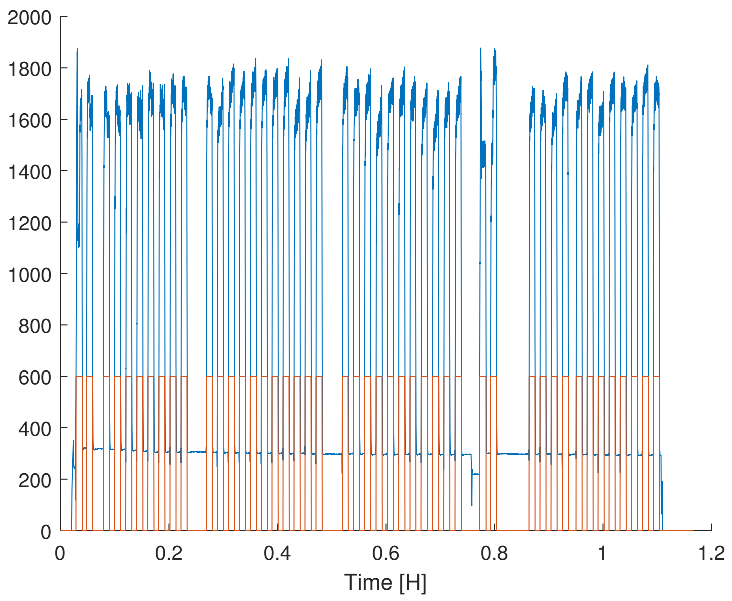


Figure 4. State detection for signal 21:20 performed on file B_02_04

Figure 5 shows pulses or cycles for Signal 5, 21:20, in each of the fifteen files. As mentioned earlier B_30_03 and B_04_04 show no pulses as all. All the pulses are around 40 seconds in duration with similar shapes but varying profiles.

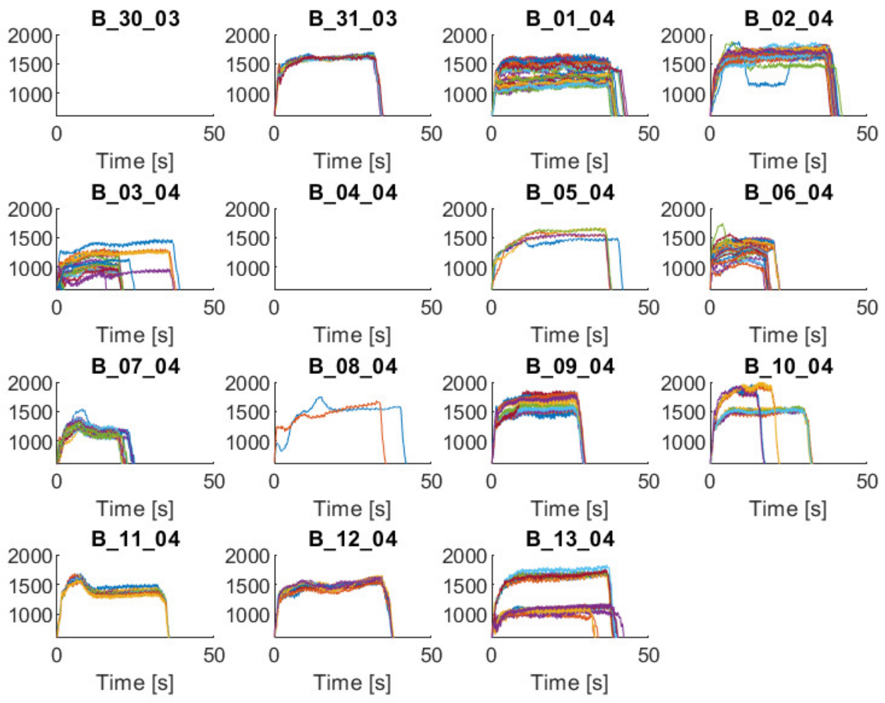


Figure 5. Identified pulses in each files for file

Figure 6 shows the signals for all 224 pulses with each sensor channel plotted in a different tile.

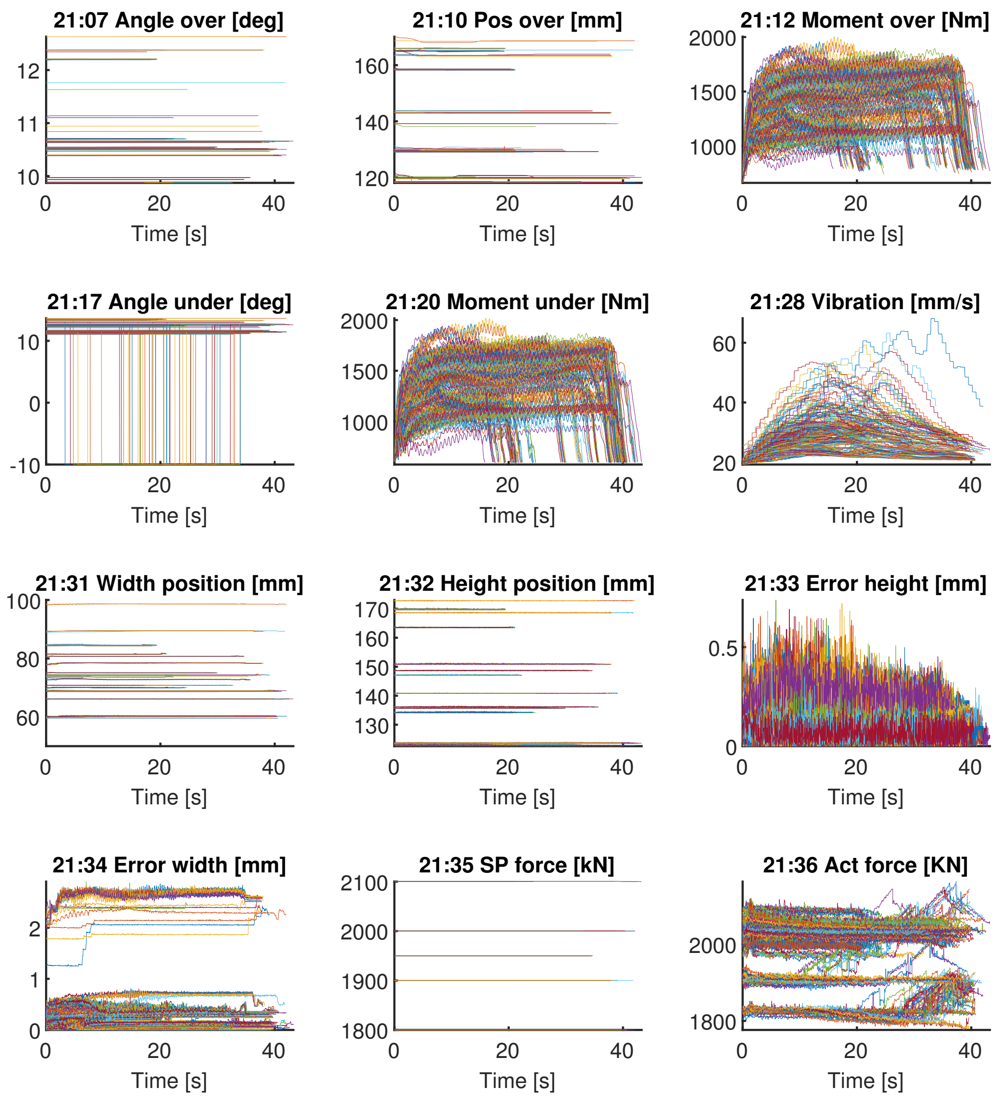


Figure 6. Signal pulses for 224 cycles on each of the 12 channels

Using the identified pulses, one can look at all the other signals during the same periods. Signal 1, 21.07 Angle over rolls and 2, 21.10 Position over rolls are fairly constant though contain some small deviations. Signal 7, 21:31 Width position and 8, 21:32 Height position are also fairly constant but contain some information when the scale is increased. Signal 21.35, Force Set Point is a 100% constant signal and contains nothing but a single value. Considering the label, it can be assumed that this is a set point that is controlled by the operator or control system and not as sensor on the machine providing feedback. Apart from using this signal for further separating the data into different categories it's likely not that useful for condition monitoring.

Figure 7 shows the state detection Signal 5, Moment under rolls, in more detail. Visually one can see these are a type of pulse single with a small amplitude frequency component. Signal 3, Moment over rolls looks like a very similar type of signal. Signals 6, 9, 10 and 11 contain some interesting information that can be used for characterize the machine state. In this context, interesting means that the signals are not constant values with minimal variance. They has different shapes, means, amplitudes, lengths, frequencies etc. so there are a number of parameters to differentiate the signals on. Constant signals can only really be differentiated in mean and length.

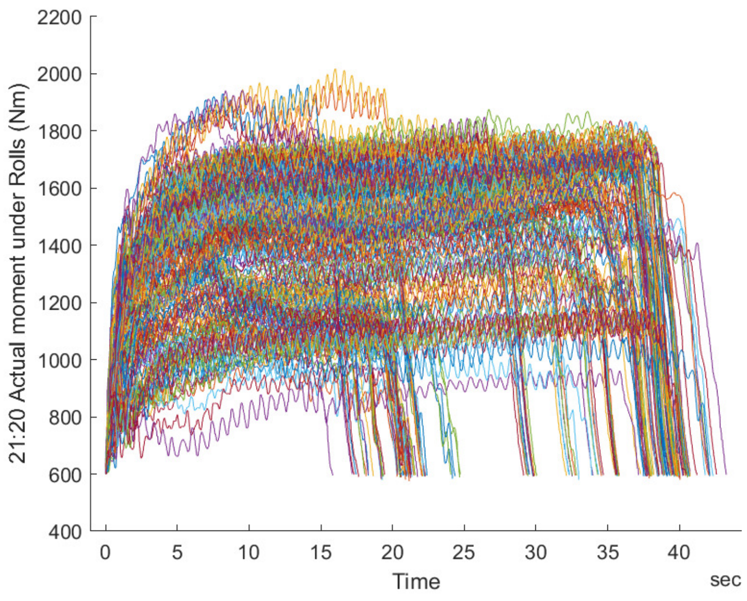


Figure 7. Signal 5 Pulses 21:20 Actual moment under rolls

5 Results

5.1 Signal Correlations

A comparison of the raw signals was performed using R_2 values. This was not performed for all files due to the long time taken to run the script, just the ones with the most pulses. Table II shows the R_2 values for each pair of signals for file 4, the day with the most pulses, with the lower triangle being a replica of the upper triangle. The column for Signal 35, Set Point Force, is minus infinity because of the perfectly constant signal. Figure 8 plots all of the signals against each other for file 4 with the R_2 values in the range of 0.2 to 0.9 printed on the plots. The colors used in the figure are used to represent the time scale with the start of the time scale being blue and trending towards red.

Starting with some basic observations from fig. 8, there are a few combinations that are extremely linear. Two pairs that are highly correlated are:

1. Signal 7, 21:31 Width position and Signal 10, 21:34 Error position for width.
2. Signal 8, 21:32 Height position and Signal 9, 21:33 Error position for height.

These data pairs appear to form a V shape with two straight lines meeting at a point. Intuitively this makes sense because the error signals themselves are fairly constant with very small variation. As the height/width position increases or decreases further away from the set point the larger or smaller the error gets. Therefore these two pairs should always be highly correlated.

Signal 3, 21:12 Actual moment over rolls and Signal 5, 21:20 Actual moment under rolls are also highly linear. This would indicate that the torque on both dimensions of the pipe are varying together.

One signal that is of interest is Signal 6, 21:28 Vibration Measurements, and two combinations it forms with Signal 3, Actual moment over rolls and Signal 5, Actual moment under rolls. Both these combinations have the same correlation value and when looking at fig. 8 there appears to be a clear non-linear relationship.

Figure 9 shows more clearly what could be an exponential or cubic relationship between the vibrations and the torque values on the upper and lower rollers. The four plots are of the four files with the most pulses and all appear to have similarities in shape. This could be explained by the fact that the larger vibration of the rollers the more torque is experienced or applied by the actuators to keep the rollers in place. Figure 9a shows what could be a time dependent relationship since the colors trend from red to blue on the right side of the plot but this is not really evident in the other plants.

While fig. 9 plots the Signal 3, Moment over rolls, against Signal 6 it should be noted that the plots for Signal 5, Moment under rolls look very similar but are not included in the report. It is also worth to remember that File 1 and 6 contain no identified pulses so the R_2 tables for those two files, while not included in the report, look quite different to the others and should be ignored.

Table II. R_2 values for each signal vs. each other signal

	07	**10**	**12**	**17**	**20**	**28**	**31**	**32**	**33**	**34**	**35**	**36**
07	1.00	0.07	0.00	0.01	0.00	0.02	0.04	0.45	0.46	0.03	-Inf	0.00
10	0.07	1.00	0.02	0.00	0.02	0.00	0.23	0.15	0.14	0.19	-Inf	0.00
12	0.00	0.02	1.00	0.02	1.00	0.47	0.02	0.00	0.00	0.01	-Inf	0.01
17	0.01	0.00	0.02	1.00	0.02	0.22	0.00	0.00	0.00	0.00	-Inf	0.01
20	0.00	0.02	1.00	0.02	1.00	0.46	0.01	0.00	0.00	0.01	-Inf	0.01
28	0.02	0.00	0.47	0.22	0.46	1.00	0.02	0.01	0.01	0.03	-Inf	0.00
31	0.04	0.23	0.02	0.00	0.01	0.02	1.00	0.17	0.15	0.99	-Inf	0.04
32	0.45	0.15	0.00	0.00	0.00	0.01	0.17	1.00	0.99	0.14	-Inf	0.00
33	0.46	0.14	0.00	0.00	0.00	0.01	0.15	0.99	1.00	0.13	-Inf	0.00
34	0.03	0.19	0.01	0.00	0.01	0.03	0.99	0.14	0.13	1.00	-Inf	0.04
35	-0.00	-0.00	-0.00	0.00	-0.00	0.00	-0.00	-0.00	0.00	0.00	-Inf	-0.00
36	0.00	0.00	0.01	0.01	0.01	0.00	0.04	0.00	0.00	0.04	-Inf	1.00

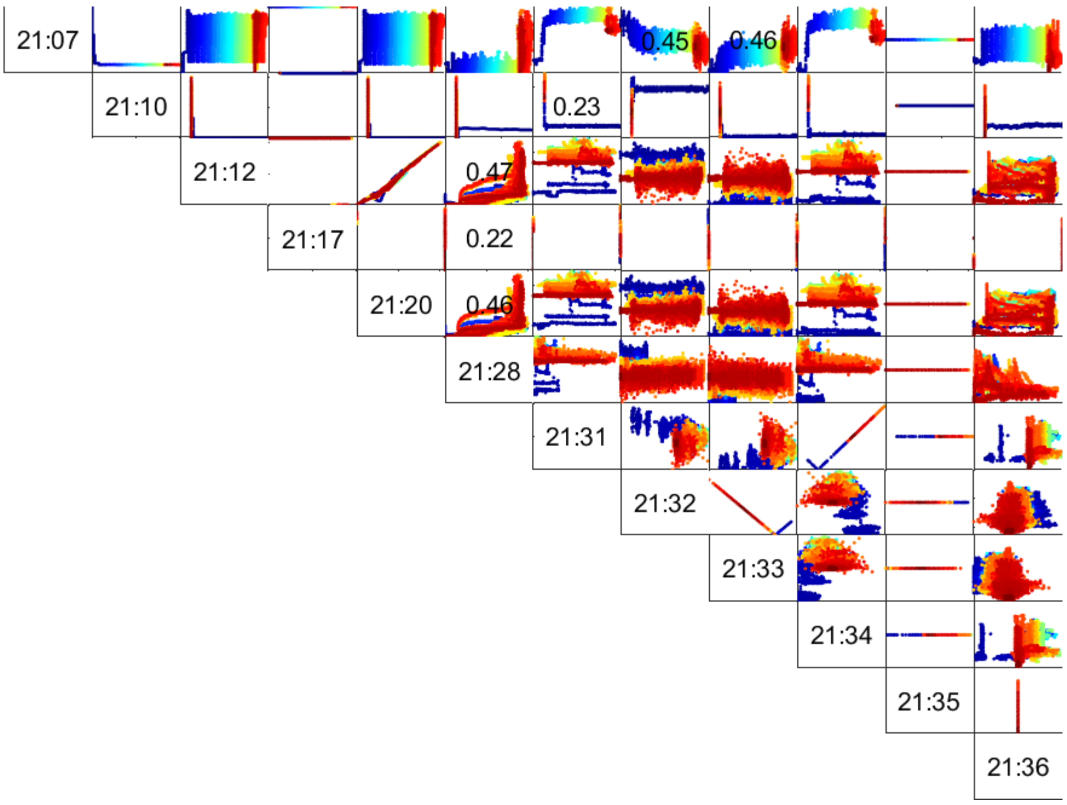
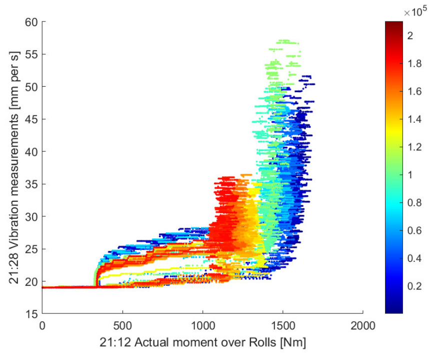
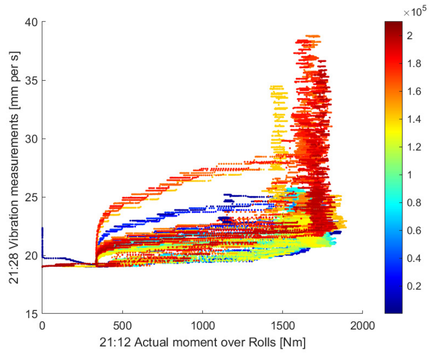


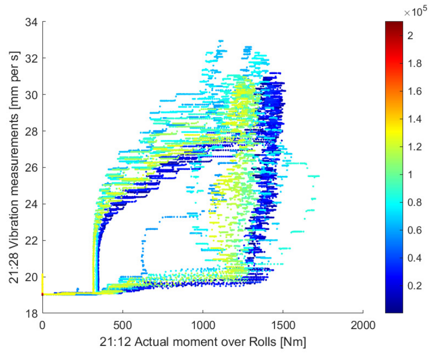
Figure 8. Plot of each signal vs every other signal with R_2 values between 0.2 and 0.9 for File 4



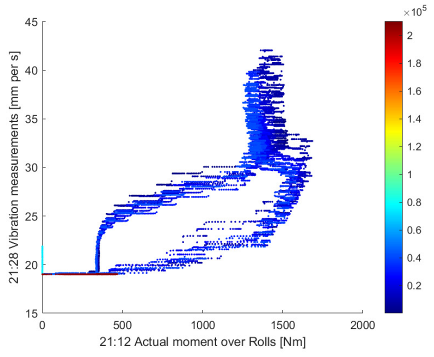
(a) File 3



(b) File 4



(c) File 8



(d) File 13

Figure 9. Vibration vs Moment where color equates to time over approximately 1 hour

5.2 Feature Selection

Once the data has been pre-processed the feature values are calculated for each identified production cycle. For every channel, the R_2 value for every pair of signal features is computed and the pairs of data points for each cycle are plotted against each other in a scatter plot. This is used as a first stage for filtering out irrelevant data. A small, or zero, R_2 value indicates that there is a very weak or no linear relationship between that pair of features. On the other hand an R_2 value of close to 1 indicates that the two features are perfectly or close to perfectly linear.

Two threshold values are used to select a range of R_2 correlated features with this method. An upper and and a lower R_2 threshold value are specified which reveals feature pairs which are somewhat linear or varying in nature. These are the combinations that are of interest to look for condition indicators.

A semi automated method is be used to narrow down on feature pairs of interest. A script cycles through the table of R_2 values and the features correlated to a number of others can be isolated. Within that combination one can then remove those signals from the group which are highly correlated with each other. Not only can one compare pairs of features within the same signal but one can also look for relationships between the same feature over multiple channels. A manual approach was then used to select a few relationships of interest to include in the report.

MATLAB's Diagnostic Feature Designer was used calculate the features from the signal pulse data with Table III showing the features that were extracted. Throughout the remainder of the report some figures are labeled with a feature number instead of the full text due to readability and space constraints. Use Table III to refer to which feature the indexes represents.

Table III. Feature Names

1	Clearance Factor	9	SINAD
2	Crest Factor	10	Shape Factor
3	Impulse Factor	11	Skewness
4	Kurtosis	12	Standard Deviation
5	Mean	13	THD
6	Peak Value	14	Band Power
7	RMS	15	Peak Amplitude 1
8	SNR	16	Peak Frequency 1

5.3 Feature vs Feature Models

Figure 10 shows an example of plotting all features against one another. This was repeated for all signals, one is shown in the main section of this report, the rest can be found in Appendix B. Each number on the diagonal represents a feature from Table III, starting with the signal features followed by the spectral features. Only the R_2 values between 0.3 and 0.8 are printed on the relevant plots. The colors used in these plots are used to represent a pseudo time scale. Since the data is only from 1 hour per day the color spectrum is non linear however it may be useful in revealing any trends over time.

Once a subset of features are chosen, equation 15 is used to create a model. If two features are selected, for example kurtosis and mean, then the model is given by

$$\mathbf{X}_{\text{kurtosis}} = \boldsymbol{\theta}'\mathbf{X}_{\text{mean}} \tag{18}$$

where $\mathbf{X}_{\text{kurtosis}}$ is an $(N \times 1)$ array, $\boldsymbol{\theta}$ is a (2×1) array and $\mathbf{X} = \begin{bmatrix} 1 & \mathbf{X}_{\text{mean}}(1) \\ \vdots & \vdots \\ 1 & \mathbf{X}_{\text{mean}}(N) \end{bmatrix}$, which is an $(N \times 2)$ matrix.

It can be noted in fig. 10 that Features 1, 2 and 3 all have very straight lines so here we can say that these features (Clearance, Crest and Impulse Factor), are linearly correlated and thus need only include one of these in any model. This is the same situation for Features 5, 6 and 7 (Mean, Peak Value and RMS). Looking at Feature 14, Band Power, it can be seen that the R_2 value is within the specified range when compared with Features 10, Shape Factor and 12, Standard Deviation. Figure 11a shows these three features plotted against one another for all cycles. As expected they are somewhat similar in pattern but not exactly the same. Linear regression models were created, figs. 11b and 11c with the NRMSE value shown in the plots. Figure 11d shows a three dimensional model for all three of those Features which looks extremely linear apart from a few outlines. The NRMSE value is increased from the individual plots which means it's a pretty good fit. It's also worth to note, while the figures are not included in the report, Signal 5 21:20 Actual moment over rolls has the same characteristics of Signal 3, Actual moment under rolls which means one can not make an assumption that the variance is constant along the signal.

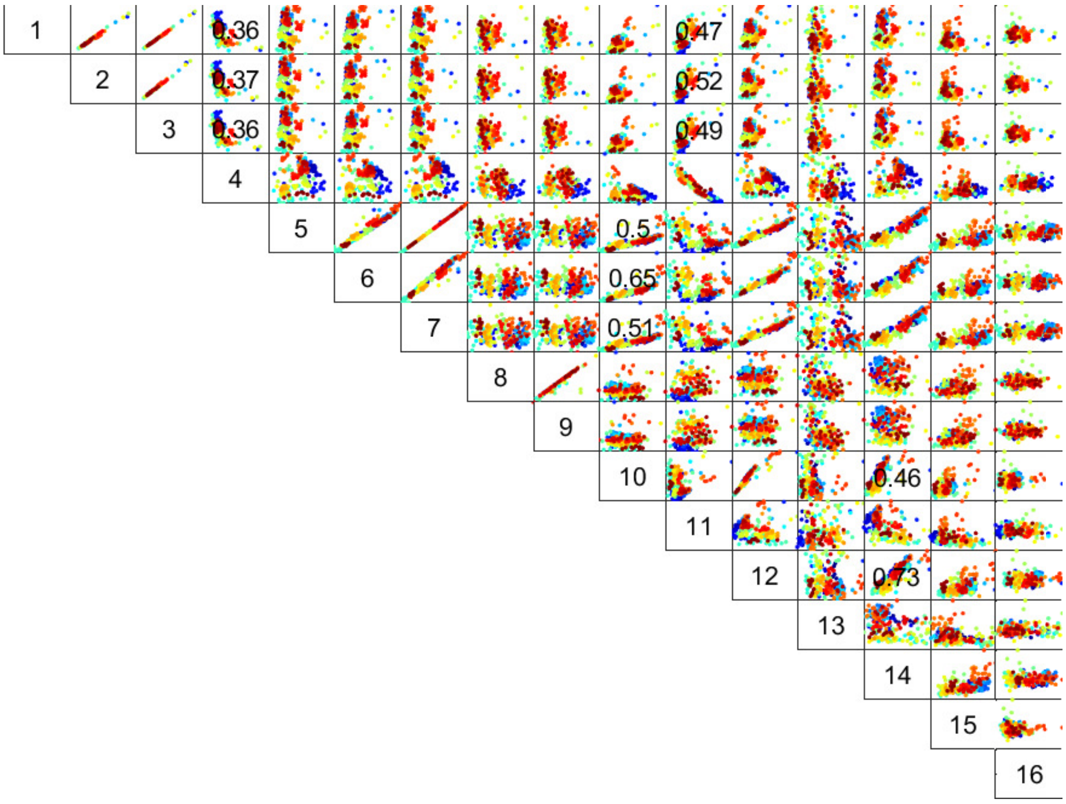
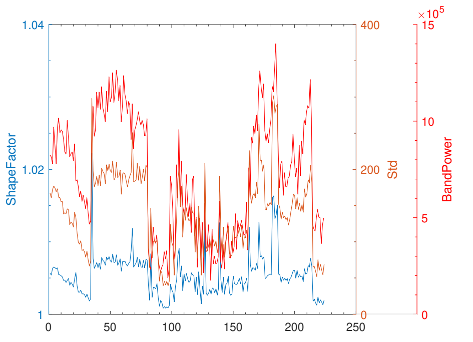
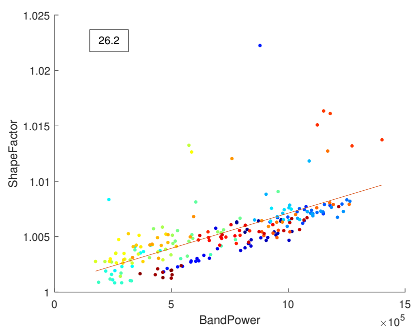


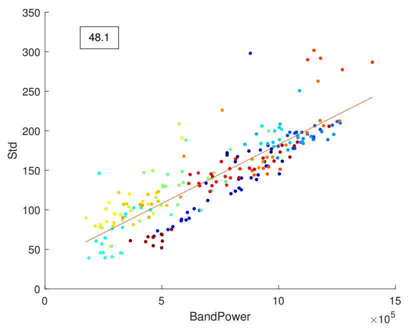
Figure 10. Feature vs Feature for Signal 3, 21:12 Actual moment over rolls



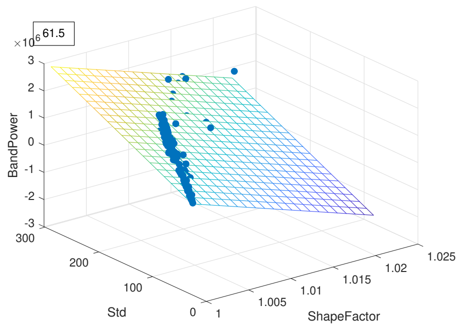
(a) Feature values over time: Shape Factor (blue), Standard Deviation (blue) and Band Power (red).



(b) Band power vs Shape factor with data points in color ordered in time and linear model in red with NRMSE = 26.2.



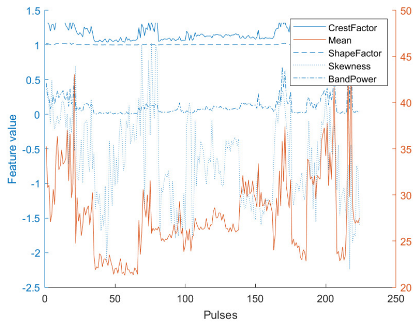
(c) Band power vs Standard Deviation with data points in color ordered in time and linear model in red with NRMSE = 48.1.



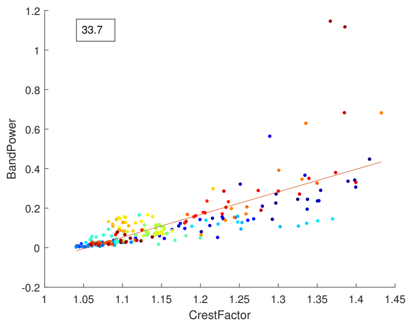
(d) Multivariate Model of Shape Factor, Standard Deviation and Band Power with NRMSE = 61.5.

Figure 11. Signal 3, 21:12 Actual moment over rolls

The vibration features in fig. 12 shows an interesting relationship. Five of the features are have correlation values within the specified range but one combination in particular has a different relationship to the others. The Crest Factor vs Band Power exhibits a relationship with an increasing variance.



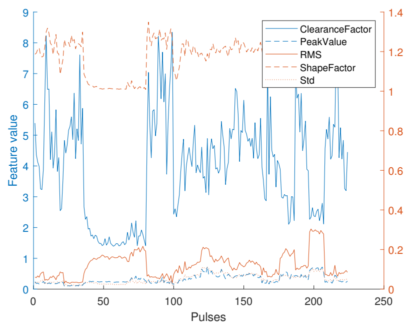
(a) Feature values over time: Crest Factor (blue), Mean (red), Shape Factor (blue dashed), Skewness (blue dotted) and Band power (blue dash-dotted).



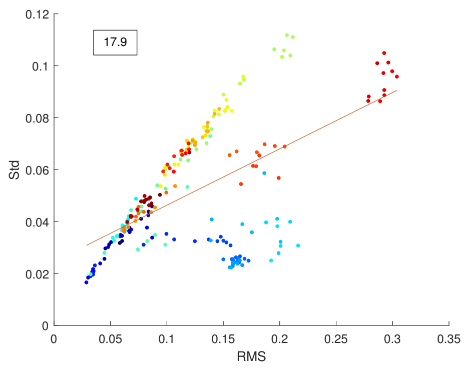
(b) Crest factor vs Band power with data points in color ordered in time and linear model in red with NRMSE = 33.7.

Figure 12. Signal 6, 21:28 Vibration measurements

Looking at the error for the height position, there is an an interesting pattern in the data for Feature 7, RMS. Figure 13a shows a number of the correlated features drawn on the same plot however the combination of RMS and Standard Deviation is the one of interest. Figure 13b shows these two features plotted against each other and there is a straight line clearly apparent however some other little clusters appear adjacent to this line. The straight line could represent when the machine is working in a good operating window and the clusters when it deviates from this.



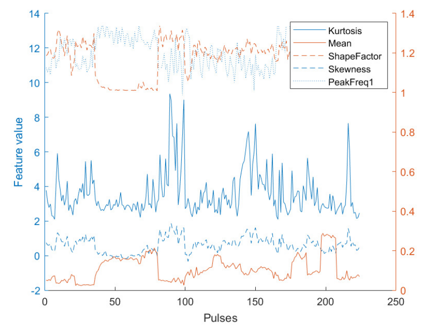
(a) Feature values over time: Clearance Factor (blue), Peak Value (dotted blue), RMS (red), Shape Factor (dashed red) and Standard Deviation (dotted red).



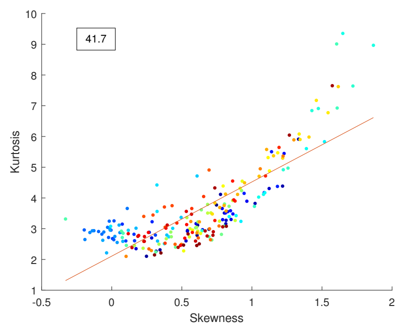
(b) RMS vs Standard Deviation with data points in color ordered in time and linear model in red with NRMSE = 17.9.

Figure 13. Signal 9, 21:33 Error position for height

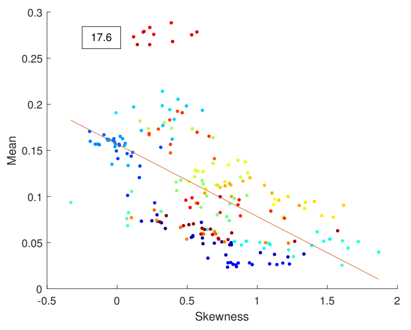
Figure 14 shows the subset of features for Feature 11, Skewness. Two combinations appear to show exponential relationships, Skewness vs Kurtosis fig. 14b and Skewness vs Shape Factor fig. 14d.



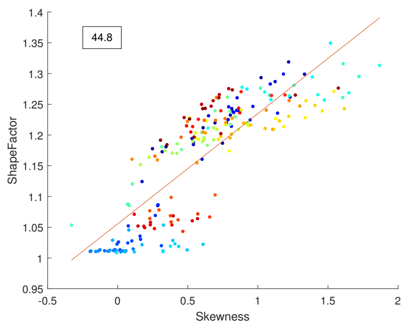
(a) Feature values over time: Kurtosis (blue), Mean (red), Shape Factor (dashed red), Skewness (dashed blue) and Peak Frequency (dotted blue).



(b) Skewness vs Kurtosis with data points in color ordered in time and linear model in red with NRMSE = 41.7.



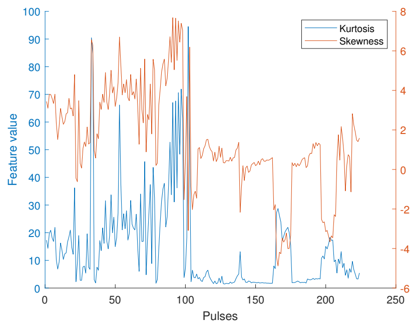
(c) Skewness vs Mean with data points in color ordered in time and linear model in red with NRMSE = 17.6.



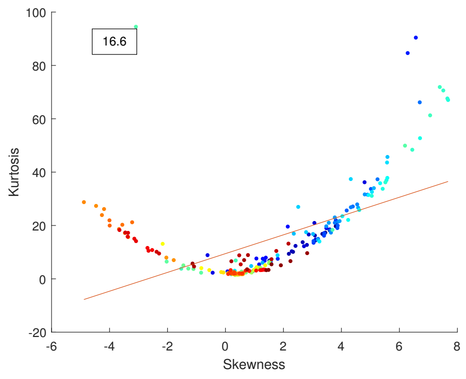
(d) Skewness vs Shape Factor with data points in color ordered in time and linear model in red with NRMSE = 44.7.

Figure 14. Signal 9, 21:33 Error position for height

Figure 15 plots two features for Signal 10, Width error. Figure 15b shows a clear non-linear parabolic relationship between the Skewness and the Kurtosis. There could also be a time varying relationship here since the left side of the data set appears to be mostly red and moves to predominantly blue as the skewness increases.



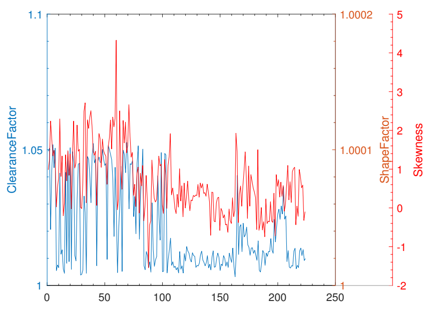
(a) Feature values over time: Kurtosis (blue) and Skewness (red).



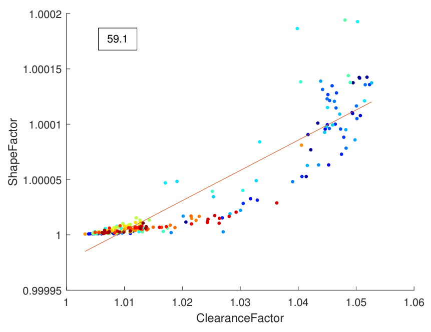
(b) Skewness vs Kurtosis with data points in color ordered in time and linear model in red with NRMSE = 16.6.

Figure 15. Signal 10, 21:34 Error position for width

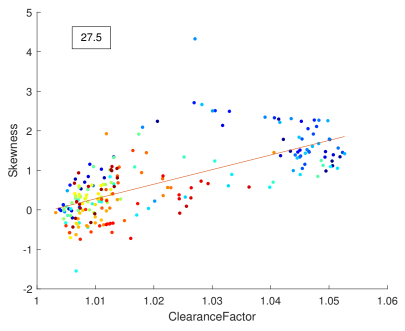
Figure 16 shows two relationships of interest. Figure 16b appears to be a non-linear exponential relationship where the variance is not constant, but increasing as both values move away from 1. Figure 16c shows a week linear relationship but appears to have two distinct clusters with one on either side of the plot. The 3D model in fig. 16d shows the combination of all three of these variables where there is a dense cluster of data in the lower left of the plot and some clusters of outliers higher up.



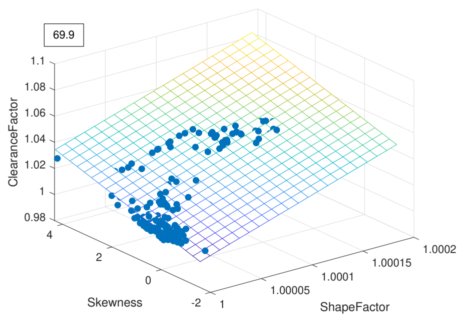
(a) Feature values over time: Clearance Factor (blue), Shape Factor (orange) and Skewness (red).



(b) Clearance Factor vs Shape Factor with data points in color ordered in time and linear model in red with NRMSE = 59.1.



(c) Clearance vs Skewness Band power vs Crest factor with data points in color ordered in time and linear model in red with NRMSE = 27.5.



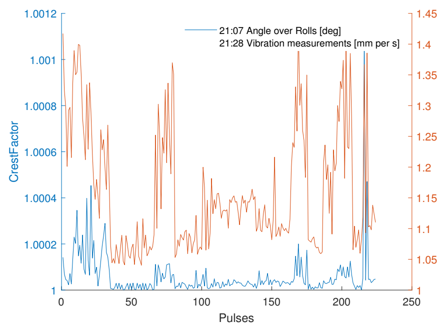
(d) Multivariate Model with Shape Factor vs Skewness vs Clearance Factor and a Fit value of 69.9.

Figure 16. Signal 12, 21:36 Actual force

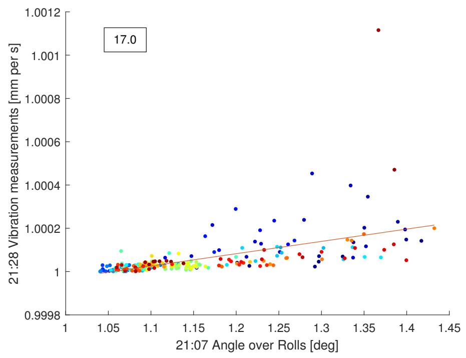
5.4 Signal vs Signal Models

Just as in the last section, where each feature was plotted against every other feature within the same signal, the same thing can be done within a feature. For example we can take the mean of all sensor channels and plot these against each other.

One feature that was chosen to include in the report is the Crest Factor, fig. 17. Figure 17a shows Signal 1 21:07 Angle over rolls vs Signal 6 21:28 Vibrations where the linear model is very accurate close to the Y axis with increasing variance as the Crest Factor of the Angle Over rolls increases.



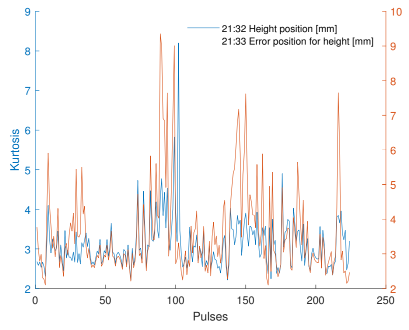
(a) Feature values over time: Clearance Factor (blue), Shape Factor (orange) and Skewness (red).



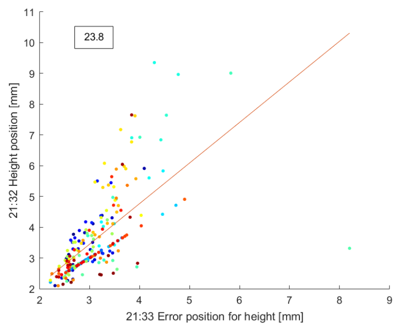
(b) Angle over rolls vs Vibration with data points in color ordered in time and linear model in red with NRMSE = 17.

Figure 17. Feature 2 Crest Factor

Figure 18 shows the Kurtosis values for the Height vs the Height Error. It is interesting to note in fig. 18b the there is again increasing variance as the the values increase, but also that there appears to be a cluster of data the forms a very straight line in the lower portion of the graph. This could indicate a certain operating condition and the rest of the data indicates something else. Without maintenance data it is not known whether this is normal operating condition or not. This linear model is valid for errors in Height between 2 mm to 6 mm and height position between 2 to 10mm but outside of these this is not enough data.



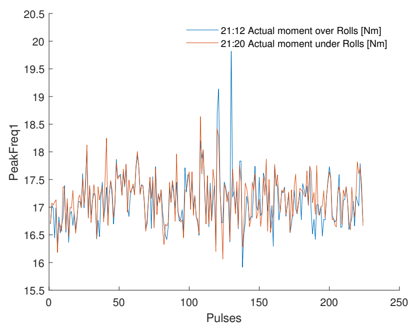
(a) Feature values over time: Clearance Factor (blue), Shape Factor (orange) and Skewness (red).



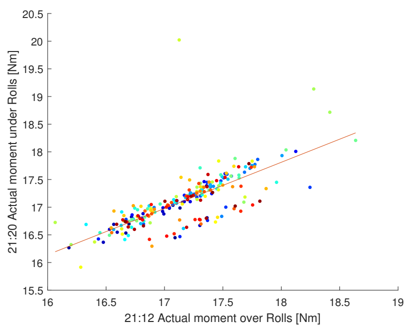
(b) Height Error vs Height Position with data points in color ordered in time and linear model in red with NRMSE = 23.8.

Figure 18. Feature 4 Kurtosis

Figure 19b shows the peak frequency of the Actual Moment over rolls vs Actual moment under rolls. It can be noted that a majority of the data points follow the a linear model very well. However, a subset of the available data appear to form another linear relationship with a similar gradient but slightly lower intercept.



(a) Feature values over time: Clearance Factor (blue), Shape Factor (orange) and Skewness (red).



(b) Actual moment under rolls vs Actual Moment under rolls with data points in color ordered in time and linear model in red with NRMSE = 33.6.

Figure 19. Feature 16 Peak Frequency 1

6 Discussion

This study aimed to find patterns in the data and give suggestions as to which signals and signal features are of value for future condition monitoring studies and which are not. Of the data provided, many of the signals were relatively constant showing minimal variance. In particular, Signal 11, 21:35 Set point force is perfectly constant and is not suitable for condition monitoring. The following signals are not overly useful to condition monitoring as they are fairly constant in mean value and contain little variability:

- Signal 1, 21:07 Angle over rolls
- Signal 2, 21:10 Position over rolls
- Signal 4, 21:17 Angle under rolls
- Signal 7, 21:31 Width position
- Signal 8, 21:32 Height position
- Signal 11, 21:35 Set point force

Excluding these signals from future studies already reduces the data by 50%. The remaining signals are most useful for condition monitoring:

- Signal 3, 21:12 Actual moment over rolls
- Signal 5, 21:20 Actual moment under rolls
- Signal 6, 21:28 Vibration measurements
- Signal 9, 21:33 Error position for height
- Signal 10, 21:34 Error position for the width
- Signal 12, 21:36 Actual force

The strength in this study lies in using data from an operational machine with varying scales and units. It would be worthwhile to inquire with the company about whether what they have provided is the full set of available signals. Temperature and acoustic emissions are also highly effective types of sensors for condition monitoring so if there was in fact additional data these would be of interest. If none of these sensors exist perhaps it might be a good consideration add one or more of these types of measurements to complement existing infrastructure.

The data supplied for the project is from a two week period which probably does not give a good insight into the life cycle of the machine. No maintenance data was provided. Perhaps maintenance is only required on a yearly cycle, thus the data is not fully representative of the machine degradation over time. Perhaps a better selection of data would include a snippet of data every 1-2 weeks over a year long period and thus one could get a better view of how the machine degrades over a year.

A system that has been operating for a long period of time will undergo changes due to the natural wear of the system. Ideally, data over the lifespan of a machine would be available. It is not known whether the current data is normal operating data. It could be right at the beginning of the life cycle or could be just at the end of the machines life cycle.

Throughout the report a color spectrum was used to indicate the time order of pulses in an attempt to reveal degradation conditions. It's worth remembering that this machine processes different sized tubes and thus one could confuse clustering in time with clustering in product sizes. This is where more data broken down in to different product sizes could be better at revealing degradation over time.

This study utilizes relatively simple methods to in comparison to the types of methods mentioned in earlier sections of the report. An analysis of such data can be expanded exponentially but one can only look at so much. It is also beneficial to, at least begin with, simple things like two variable linear regression and perhaps this is all that is required to identify a indicator of degradation state.

The subject matter, methods and results used in this report are inline with target 9.5 of the 2030 Agenda for sustainable development[30]. Hopefully this work will upgrade technology capabilities in industrial sectors and will assist workers in the field of research and development.

6.1 Future Work

This report has identified patterns in the data and potential clusters based on visual observations. However, no attempts were made to implement any clustering algorithms. This would be the logical next step, to apply machine learning techniques, in trying to identify degradation and failure states.

The straightening machine in this study produces tubes of different diameters. At least one of the signals, Height and/or Width Position, can be used to indicate the diameter. This could be used to classify the data further into different tube widths. A suggestion for future studies is to perform an additional prepossessing step to separate data into periods where the machine is producing tubes with certain diameters. Some of the clustering suggestions or relationships given in this study could just be identifying when the machine is producing a tube of a certain width. Separating data into pipe widths and focusing on only one tube width would remove any chance of this. A method like this would have been challenging since only 224 pulses were identified to begin with and thus a further reduction might result in not enough data to make meaningful observations.

As mentioned previously, the data for this study was taken over a two week period with no data on maintenance. For the next set of data might it might be beneficial to request periods of data with longer intervals between them. Ideally, obtaining some data before a known maintenance task and after might assist in identify clusters which are indicative of the machine degradation state.

While this report highlighted some non-linear relationship it made no attempt at modeling them. This could be a worthwhile endeavor for the next iteration of work.

7 Conclusion

A study of the available data on a steel tube straightening machine was conducted with an eye to using the existing infrastructure for condition monitoring. R_2 correlation was useful for identifying relationships between the sensor channels and between features. Range thresholds of around 0.2 to 0.9 were a good choice for narrowing down the signals to a manageable number.

The most promising signals for condition monitoring are the two torque moments (under and over rolls), the two error positions (height and width), the vibration measurement and the actual force. This already reduces the data by 50%. No one feature stood out above all others for condition monitoring suitability.

Specific relationships recommended for further investigations are:

- RMS vs Standard Deviation of Signal 9, 21:33 Error position for height.
- Skewness vs Kurtosis of Signal 10, 21:34 Error position for width.
- Clearance Factor vs Shape Factor of Signal 12, 21:36 Actual Force.
- Moment under vs Moment over rolls for Feature 16, Peak Frequency 1.

Potential non-linear relations were identified in:

- Signal 10, 21:34 Error position for width - Skewness vs Kurtosis.
- Signal 12, 21:36 Actual force - Clearance Factor vs Shape Factor.

8 References

[1] T. Buckley, B. Ghosh, and V. Pakrashi, "A feature extraction & selection benchmark for structural health monitoring," *Structural Health Monitoring*, vol. 22, no. 3, pp. 2082–2127, 2023.

[2] S. Telford, M. I. Mazhar, and I. Howard, "Condition based maintenance (cbm) in the oil and gas industry: An overview of methods and techniques," in *Proceedings of the 2011 international conference on industrial engineering and operations management, Kuala Lumpur, Malaysia*, 2011, pp. 22–24.

[3] L. Tolosi and T. Lengauer, "Classification with correlated features: Unreliability of feature ranking and solutions," *Bioinformatics*, vol. 27, no. 14, pp. 1986–1994, 2011.

[4] M. Tiboni, C. Remino, R. Bussola, and C. Amici, "A review on vibration-based condition monitoring of rotating machinery," *Applied Sciences*, vol. 12, no. 3, p. 972, 2022.

[5] D. Kateris, D. Moshou, X.-E. Pantazi, I. Gravalos, N. Sawalhi, and S. Loutridis, "A machine learning approach for the condition monitoring of rotating machinery," *Journal of Mechanical Science and Technology*, vol. 28, pp. 61–71, 2014.

[6] B. Zhang, L. Zhang, and J. Xu, "Degradation feature selection for remaining useful life prediction of rolling element bearings," *Quality and Reliability Engineering International*, vol. 32, no. 2, pp. 547–554, 2016.

[7] M. Kato, A. Hasegawa, S. Sugyo, H. Nakamura, M. Kobayashi, and Y. Morimoto, "Straightening technology of round bars using 2-roll rotary straightener," *Procedia Engineering*, vol. 81, pp. 233–238, 2014.

[8] L.-d. Ma, Y.-k. Du, Z.-j. Meng, L.-f. Ma, Z.-j. Liu, and P.-y. Liu, "Effect of process parameters on steel tube roundness in straightening process," *Journal of Iron and Steel Research International*, vol. 27, pp. 1270–1283, 2020.

[9] L. Ma, Z. Liu, L. Ma, and Y. Du, "Analysis of eleven cross-roll straightening process of steel tube based on cubic spline function and continuous bending elastic-plastic theory," *The International Journal of Advanced Manufacturing Technology*, vol. 112, pp. 3235–3245, 2021.

[10] G. Yu, R. Zhai, J. Zhao, and R. Ma, "Theoretical analysis and numerical simulation on the process mechanism of two-roller straightening," *The International Journal of Advanced Manufacturing Technology*, vol. 94, pp. 4011–4021, 2018.

[11] N. Das Talukder and A. Singh, "Mechanics of bar straightening, part 1: General analysis of straightening process," 1991.

[12] H. Yoshimura, H. Demiya, and Y. Mihara, "Effect of contacting ratio on roundness in straightening process of steel tube," *Tetsu to Hagane-Journal of the Iron and Steel Institute of Japan*, vol. 95, no. 11, pp. 801–806, 2009.

[13] Z. Zhang, "Modeling and simulation for cross-sectional ovalization of thin-walled tubes in continuous rotary straightening process," *International Journal of Mechanical Sciences*, vol. 153, pp. 83–102, 2019.

[14] Z. Zhang, Z. Wu, F. Liu, and J. Song, "Study of the hydraulic system simulation and pid autotuning based on relay feedback of the tantalum-niobium tube straightener," in *2010 5th IEEE Conference on Industrial Electronics and Applications*, IEEE, 2010, pp. 1253–1257.

[15] W. Caesarendra and T. Tjahjowidodo, "A review of feature extraction methods in vibration-based condition monitoring and its application for degradation trend estimation of low-speed slew bearing," *Machines*, vol. 5, no. 4, p. 21, 2017.

[16] S. Adams, R. Meekins, P. A. Beling, *et al.*, "A comparison of feature selection and feature extraction techniques for condition monitoring of a hydraulic actuator," in *Annual Conference of the PHM Society*, vol. 9, 2017.

[17] S. Hong, Z. Zhou, E. Zio, and K. Hong, "Condition assessment for the performance degradation of bearing based on a combinatorial feature extraction method," *Digital Signal Processing*, vol. 27, pp. 159–166, 2014.

[18] R. Teti, K. Jemielniak, G. O'Donnell, and D. Dornfeld, "Advanced monitoring of machining operations," *CIRP annals*, vol. 59, no. 2, pp. 717–739, 2010.

[19] A. Soualhi, B. El Yousfi, H. Razik, and T. Wang, "A novel feature extraction method for the condition monitoring of bearings," *Energies*, vol. 14, no. 8, p. 2322, 2021.

[20] G. D'Emilia, A. Gaspari, and E. Natale, "Physical and metrological approach for feature's definition and selection in condition monitoring," *Sensors*, vol. 19, no. 23, p. 5186, 2019.

[21] H. Ahmed and A. K. Nandi, *Condition monitoring with vibration signals: Compressive sampling and learning algorithms for rotating machines*. John Wiley & Sons, 2020.

[22] H. Wang, Y. Tang, and R. Hong, "Multiple physical signals based residual life prediction model of slewing bearing," *Journal of Vibroengineering*, vol. 18, no. 7, pp. 4340–4353, 2016.

[23] T. M. Inc., *Diagnostic feature designer (r2022b)*, Natick, Massachusetts, United States, 2022. [Online]. Available: `https://www.mathworks.com`.

[24] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*. Springer, 2013, vol. 112.

[25] C. Lu, J. Chen, R. Hong, Y. Feng, and Y. Li, “Degradation trend estimation of slewing bearing based on lssvm model,” *Mechanical Systems and Signal Processing*, vol. 76, pp. 353–366, 2016.

[26] A. Malhi and R. X. Gao, “Pca-based feature selection scheme for machine defect classification,” *IEEE transactions on instrumentation and measurement*, vol. 53, no. 6, pp. 1517–1525, 2004.

[27] O. B. Gonzalez and D. Rönnow, “Time series modelling of a radial-axial ring rolling system,” *International Journal of Modelling, Identification and Control*, vol. 43, no. 1, pp. 13–25, 2023.

[28] S. Martín del Campo Barraza, “Unsupervised feature learning applied to condition monitoring,” Ph.D. dissertation, Luleå University of Technology, 2017.

[29] iba AG. “Ibam-daq.” (2023), [Online]. Available: `https://www.iba-ag.com/en/ibam-daq` (visited on 08/29/2023).

[30] U. Nations, “Department of economic and social affairs sustainable development. the 17 goals,” 2015.

Appendix A   Matlab Code

```
%{
    Date: 2023/08/23
    Filename: LoadRawData.m
    Author: Paul Barron
    Description: Loads the raw data from files and create labels for sensor
    channels
%}
%%
clear;
close all;
warning off;
saveFigures = false;

Ts = 0.02; % Sampling interval for data
fs = 1 / Ts;
dataSubfolder = "../SMT_data_20220520/";

filenames = [ "B_30_03" "B_31_03" "B_01_04" "B_02_04" ...
              "B_03_04" "B_04_04" "B_05_04" "B_06_04" ...
              "B_07_04" "B_08_04" "B_09_04" "B_10_04" ...
              "B_11_04" "B_12_04" "B_13_04" ];
numOfFiles = size(filenames, 2);

% Names of the signals in the data
sensorNames = [ "21:07 Angle over Rolls [deg]"...
                "21:10 Position over Rolls [mm]"...
                "21:12 Actual moment over Rolls [Nm]"...
                "21:17 Angle under roll [deg]"...
                "21:20 Actual moment under Rolls [Nm]"...
                "21:28 Vibration measurements [mm per s]"...
                "21:31 Width position [mm]"...
                "21:32 Height position [mm]"...
                "21:33 Error position for height [mm]"...
                "21:34 Error position for the width [mm]"...
                "21:35 Set point force [kN]"...
                "21:36 Actual force [kN]"];
numOfSignals = size(sensorNames, 2);

% Names of the signals in the data
sensorUnits = [ "Angle over [deg]"...
                "Pos over [mm]"...
                "Moment over [Nm]"...
                "Angle under [deg]"...
                "Moment under [Nm]"...
                "Vibration [mm/s]"...
                "Width position [mm]"...
                "Height position [mm]"...
                "Error height [mm]"...
                "Error width [mm]"...
                "SP force [kN]"...
                "Act force [KN]"];
```

```
function [pulses, mask, flag, maxPulseLength] = fnStateDetection(
    dataArrayLocal, threshold, minPulseTimeThreshold, signalNumber)
 %{
    Date: 2023/03/19
    Filename: fnStateDetction.m
    Author: Paul Barron
    Description:
    pulses: numOfPulses x numOfSignals x pulseNumOfSamples
    mask:   array that is 1 x length (numOfSamples)
    flag:   array that is 1 x numOfPulses, true = number of samples in pulse
         is less than min threshold to be considred a real pulse
    maxPulseLength: integer for the max number of samples in the longest
        pulse
%}

    % Extract signal #5 (Signal 20)
    signal20 = dataArrayLocal(signalNumber,2);
    signal20 = cell2mat(signal20);

    % Extract number of samples and sensors
    [numOfDataSamples, ~] = size(signal20);
    [numOfSensors, ~] = size(dataArrayLocal);
    mask = zeros(1, numOfDataSamples);

    % Loop through the data and record transitions from below 600 to above
    % and visa versa
    startValues = [];
    endValues = [];
    startLow = false;
    for i=2:numOfDataSamples
        if signal20(i) > threshold
            mask(i) = threshold;
        end

        if signal20(i) > threshold & signal20(i-1) <= threshold
            startValues(end+1) = i;
            startLow = true;
        end

        if signal20(i) < threshold & signal20(i-1) >= threshold & startLow
            endValues(end+1) = i;
        end
    end


    [~, numOfPulses] = size(startValues); % Calculate number of pulses
        depending on number of positive transitions
    pulses = {};
    for i = 1:numOfPulses
        signalArray = zeros(endValues(i)-startValues(i)+1, 12);
        for j = 1:numOfSensors
            arrayData = cell2mat(dataArrayLocal(j,2));
            columnToAdd = arrayData( startValues(i):endValues(i) );
            signalArray(:,j) = columnToAdd;
        end
        pulses{i} = signalArray;
    end


    % Check whether we think any of the pulses are not valid
    pulseLengths = endValues - startValues;
    flag = false(numOfPulses);
    medianPulseLength = median(pulseLengths);
    for i = 1:numOfPulses
        sprintf("Median %d Length: %d", medianPulseLength, pulseLengths(i));
        if pulseLengths(i) < minPulseTimeThreshold
            flag(i) = true;
        end
    end

    if (size(pulseLengths) <= 0)
        maxPulseLength = 0;
    else
        maxPulseLength = max(pulseLengths);
    end
end
```

```
%{
    Date: 2023/08/23
    Filename: PreprocessData.mlx
    Author: Paul Barron
    Description: This script performs the state detection to extract the
        pulse data. It calls uses the function fnStateDetection which does
        the processing. Here the data is combined into in an array and the
        pulses that are too short to be real pulses are discarded.
%}
%%
saveFigures = false;
pulseThreshold = 600; % Threshold for state detection given by company
minPulseTimeThreshold = 500; % Samples

% Loop through the files, load the data the perform the state detection
% Extract the pulses for each sensor signal based on Signal 20
signalNumberForDetection = 5;
signalNumberToPlot = 5; % This is signal 21:20
[~,numOfFiles] = size(filenames);
mask = {numOfFiles};
flag = {numOfFiles};
pulseData = {numOfFiles};
rawDataArray = {numOfFiles};
totalPulses = 1;
numOffalsePulses = 0;

for fileIndex=1:numOfFiles
    % Load the data for the current file
    rawDataStruct =  load(dataSubfolder + filenames(fileIndex) + ".mat");
    rawDataArray{fileIndex} = struct2array(rawDataStruct);

    %pulses is a 2D array with dimensions (1 * numOfPulses)
    [pulseData{fileIndex}, mask{fileIndex}, flag{fileIndex}, maxPulseLengths
        (fileIndex)] = fnStateDetection(rawDataArray{fileIndex},
        pulseThreshold, minPulseTimeThreshold, signalNumberForDetection);

    % Plot the extracted Signal 20 pulses for this particular file
    subplot(4, 4, fileIndex);
    hold on;
    [~, numOfPulses] = size(pulseData{fileIndex});
    for pulseIndex=1:numOfPulses
        x = pulseData{fileIndex}{pulseIndex}(:, signalNumberToPlot);
        t = 0 : Ts : (Ts * size(x, 1) - Ts);
        plot(t, x);
    end
    title(filenames(fileIndex), 'Interpreter', 'none');
    xlabel("Time [s]");
    axis([0 50 pulseThreshold 2000])

    % Combine the pulse data for this file with all the others
    for pulseIndex = 1:numOfPulses
        if flag{fileIndex}(pulseIndex) == false % Only include pulses that aren't marked as potential false ones
            for sensorIndex = 1:numOfSignals % Add data for each sensor
                tempData = pulseData{fileIndex}{pulseIndex}(:, sensorIndex);
                pulseData_Timetable_Standard{totalPulses, sensorIndex} = timetable(tempData, 'SampleRate', 1/Ts, 'VariableNames', sensorNames(sensorIndex));
            end
            totalPulses = totalPulses + 1;
        else
            numOffalsePulses = numOffalsePulses + 1;
        end
    end
end
%%
if saveFigures
    print -depsc '..\Latex Document\figures\IdentifiedPulsesFig';
end
%%
%% Remove DC Offset
%% Pad to the length of longest pulse
maxPulseLength = max(maxPulseLengths) + 1;
for fileIndex = 1:numOfFiles
    for pulseIndex = 1:size(pulseData{fileIndex},2)
        temp = pulseData{fileIndex}{pulseIndex};
        numSamplesUnderMax = maxPulseLength - size(temp, 1);
        pulseData_DcRemoved{fileIndex}{pulseIndex} = temp - mean(temp, 1);
        pulseData_Padded{fileIndex}{pulseIndex} = padarray (temp, numSamplesUnderMax, 12, "post");
        temp2 = pulseData_Padded{fileIndex}{pulseIndex};
        pulseData_DcRemovedPadded{fileIndex}{pulseIndex} = temp2 - mean(temp2, 1);
    end
end
clear temp;
clear temp2;

%% Combine the data for all padded pulses
totalPulses = 1;
for fileIndex=1:numOfFiles
    for pulseIndex = 1:size(pulseData{fileIndex},2)
        if flag{fileIndex}(pulseIndex) == false % Only include pulses that aren't marked as potential false ones
            for sensorIndex = 1:numOfSignals % Add data for each sensor
                tempData = pulseData_Padded{fileIndex}{pulseIndex}(:, sensorIndex);
                pulseData_Timetable_Padded{totalPulses, sensorIndex} = timetable(tempData, 'SampleRate', 1/Ts, 'VariableNames', sensorNames(sensorIndex));
                tempData = pulseData_DcRemoved{fileIndex}{pulseIndex}(:, sensorIndex);
                pulseData_Timetable_DcRemoved{totalPulses, sensorIndex} =
                    timetable(tempData, 'SampleRate', 1/Ts, 'VariableNames',
                     sensorNames(sensorIndex));
                tempData = pulseData_DcRemoved{fileIndex}{pulseIndex}(:,
                    sensorIndex);
                pulseData_Timetable_DcRemovedPadded{totalPulses, sensorIndex
                    } = timetable(tempData, 'SampleRate', 1/Ts, '
                    VariableNames', sensorNames(sensorIndex));
            end
            totalPulses = totalPulses + 1;
        end
    end
end
clear fileIndex;
clear pulseIndex;
clear tempData;
%% Plot DC Removed data

figure();
fig = tiledlayout(4, 4);
for fileIndex=1:numOfFiles
    nexttile; hold on;
    for pulseIndex = 1:size(pulseData_DcRemoved{fileIndex},2)
        plot(pulseData_DcRemoved{fileIndex}{pulseIndex}(:,5));
        title(pulseIndex);
    end
end
%% Plot padded data

figure();
fig = tiledlayout(4, 4);
for fileIndex=1:numOfFiles
    nexttile; hold on;
    for pulseIndex = 1:size(pulseData_Padded{fileIndex},2)
        plot(pulseData_Padded{fileIndex}{pulseIndex}(:,5));
        title(pulseIndex);
    end
end
%% Plot padded data

figure();
fig = tiledlayout(4, 4);
for fileIndex=1:numOfFiles
    nexttile; hold on;
    for pulseIndex = 1:size(pulseData_Padded{fileIndex},2)
        plot(pulseData_DcRemovedPadded{fileIndex}{pulseIndex}(:,5));
        title(pulseIndex);
    end
end
%%
clear fileIndex;
clear pulseIndex;
clear fig;
%% Plot the state detection signal vs 21:20 Actual Moment under Rollers

figure();
fig = tiledlayout(4, 4);
%title(fig, 'State detection signal 21:20 Actual Moment under Rollers');
```

```matlab
for fileIndex=1:numOfFiles
    nexttile; hold on;
    x = cell2mat(rawDataArray{fileIndex}(signalNumberToPlot, 2));
    t = 0 : Ts / 3600 : (Ts * size(x, 1) - Ts) / 3600;
    plot(t, x');
    plot(t, mask{fileIndex});
    title(filenames(fileIndex), 'Interpreter', 'none');
    xlabel("Time [H]");
    ylabel("Nm");
end
if saveFigures
    print('..\Latex Document\figures\StateDetectionFig', '-depsc');
end
filename = "..\Latex Document\figures\StateDetectionFig.png";
export_fig(filename, '-svg');
%%
figure(); hold on;
x = cell2mat(rawDataArray{4}(signalNumberToPlot, 2));
t = 0 : Ts / 3600 : (Ts * size(x, 1) - Ts) / 3600;
plot(t, x');
plot(t, mask{4});
xlabel("Time [H]");

if saveFigures
    print -depsc '..\Latex Document\figures\StateDetectionFig_B_02_04';
end
%%
numOfPulses = totalPulses - 1;
for signalIndex = 1:numOfSignals
    figure(); hold on;
    for pulseIndex=1:numOfPulses
        plot(pulseData_Timetable_Standard{pulseIndex, signalIndex}, 1, '
            LineWidth', 0.2);
    end
    filename = "..\Latex Document\figures\SignalPulse" + signalIndex;
    if (saveFigures == true)
        print(filename,'-depsc','-tiff')
    end
    filename = "..\Presentation\SignalPulse" + signalIndex;
    print(filename,'-dsvg')
end
%%
fig = tiledlayout(4, 3);
for signalIndex = 1:numOfSignals
    nexttile; hold on;
    str1 = char(sensorNames(signalIndex));
    str2 = sensorUnits(signalIndex);
    title (str1(1:5) + " " + str2);
    for pulseIndex=1:numOfPulses
        t = pulseData_Timetable_Standard{pulseIndex, signalIndex};
        tt = timetable2table(t ,'ConvertRowTimes',false);
        tVec = 0:Ts:Ts * ( size(t.Variables ,1) - 1);
        plot(tVec', tt.Variables, 'LineWidth', 0.05);
    end
    ylabel("");
    xlabel("Time [s]");
end

```

```matlab
width = 10;
height = 10;
fontsize = 6;
set(gcf,'paperunits','centimeters')
set(gcf, 'PaperPositionMode', 'manual');
set(gcf,'papersize',[width,height])
set(gcf,'paperposition',[0,0,width,height])
set(gcf, 'renderer', 'painters');
set(findall(gcf,'-property','FontSize'),'FontSize',5);
filename = "..\Latex Document\figures\SignalPulses";
if saveFigures
    print(filename,'-depsc','-tiff')
end
%%
fig = tiledlayout(3, 4);
for signalIndex = 1:numOfSignals
    nexttile; hold on;
    str1 = char(sensorNames(signalIndex));
    str2 = sensorUnits(signalIndex);
    title (str1(1:5) + " " + str2);
    for pulseIndex=1:numOfPulses
        t = pulseData_Timetable_Standard{pulseIndex, signalIndex};
        tt = timetable2table(t ,'ConvertRowTimes',false);
        tVec = 0:Ts:Ts * ( size(t.Variables ,1) - 1);
        plot(tVec', tt.Variables, 'LineWidth', 0.05);
    end
    ylabel("");
    xlabel("Time [s]");
end

width = 16;
height = 10;
fontsize = 6;
set(gcf,'paperunits','centimeters')
set(gcf, 'PaperPositionMode', 'manual');
set(gcf,'papersize',[width,height])
set(gcf,'paperposition',[0,0,width,height])
set(gcf, 'renderer', 'painters');
set(findall(gcf,'-property','FontSize'),'FontSize',5);
export_fig(filename, '-svg');
%%
fig = tiledlayout(4, 3);
for signalIndex = 1:numOfSignals
    nexttile; hold on;
    str1 = char(sensorNames(signalIndex));
    str2 = sensorUnits(signalIndex);
    title (str1(1:5) + " " + str2);
    for pulseIndex=1:numOfPulses
        t = pulseData_Timetable_DcRemoved{pulseIndex, signalIndex};
        tt = timetable2table(t ,'ConvertRowTimes',false);
        tVec = 0:Ts:Ts * ( size(t.Variables ,1) - 1);
        plot(tVec', tt.Variables, 'LineWidth', 0.05);
    end
    ylabel("");
    xlabel("Time [s]");
end

width = 10;
```

```matlab
height = 10;
fontsize = 6;
set(gcf,'paperunits','centimeters')
set(gcf, 'PaperPositionMode', 'manual');
set(gcf,'papersize',[width,height])
set(gcf,'paperposition',[0,0,width,height])
set(gcf, 'renderer', 'painters');
set(findall(gcf,'-property','FontSize'),'FontSize',5);
filename = "..\Latex Document\figures\SignalPulsesDcRemoved";
if saveFigures
    print(filename,'-depsc','-tiff')
end
%%
fig = tiledlayout(4, 3);
for signalIndex = 1:numOfSignals
    nexttile; hold on;
    str1 = char(sensorNames(signalIndex));
    str2 = sensorUnits(signalIndex);
    title (str1(1:5) + " " + str2);
    for pulseIndex=1:numOfPulses
        t = pulseData_Timetable_Padded{pulseIndex, signalIndex};
        tt = timetable2table(t ,'ConvertRowTimes',false);
        tVec = 0:Ts:Ts * ( size(t.Variables ,1) - 1);
        plot(tVec', tt.Variables, 'LineWidth', 0.05);
    end
    ylabel("");
    xlabel("Time [s]");
end

width = 10;
height = 10;
fontsize = 6;
set(gcf,'paperunits','centimeters')
set(gcf, 'PaperPositionMode', 'manual');
set(gcf,'papersize',[width,height])
set(gcf,'paperposition',[0,0,width,height])
set(gcf, 'renderer', 'painters');
set(findall(gcf,'-property','FontSize'),'FontSize',5);
filename = "..\Latex Document\figures\SignalPulsesPadded";
if saveFigures
    print(filename,'-depsc','-tiff')
end
%%
% clear x;
% clear t;
% clear signalNumberToPlot;
% clear maxPulseLength;
% clear maxPulseLengths;
% clear minPulseTimeThreshold;
% clear fig;
% clear fileIndex;
% clear numSamplesUnderMax;
% clear tVec;
% clear width;
% clear tt;
% clear height;
% clear flag;
%%
```

```
[FullPath,Filename,ext]=fileparts(matlab.desktop.editor.getActiveFilename);
currentFile = strcat(Filename, ext);
path = export(currentFile, Format="m", OpenExportedFile=false);
```

```
%{
    Date: 2023/08/23
    Filename: LoadFeatures.m
    Author: Paul Barron
    Description: This script loads the features calculated by Matlab's
        Diagnostic Feature Designer
%}

% Time Tables are equivelant to the Signal Numbers i.e. TimeTable = 21.07
signalNames_Time = [
                "TimeTable_sigstats/"...
                "TimeTable1_sigstats/"...
                "TimeTable2_sigstats/"...
                "TimeTable3_sigstats/"...
                "TimeTable4_sigstats/"...
                "TimeTable5_sigstats/"...
                "TimeTable6_sigstats/"...
                "TimeTable7_sigstats/"...
                "TimeTable8_sigstats/"...
                "TimeTable9_sigstats/"...
                "TimeTable10_sigstats/"...
                "TimeTable11_sigstats/"
              ];
numOfSignals = size(signalNames_Time, 2);

% Time domain features produced by Matlab DFD
featureNames_Time = [
                "ClearanceFactor"... %1
                "CrestFactor"... %2
                "ImpulseFactor"... %3
                "Kurtosis"... %4
                "Mean"... %5
                "PeakValue"... %6
                "RMS"... %7
                "SNR"... %8
                "SINAD"... %9
                "ShapeFactor"... %10
                "Skewness"... %11
                "Std"... %12
                "THD" %13
               ];
numOfFeatures_Time = size(featureNames_Time, 2);

% Time Tables are equivelant to the Signal Numbers i.e. TimeTable = 21.07
signalNames_Freq = [
                "TimeTable_ps_spec/"...
                "TimeTable1_ps_spec/"...
                "TimeTable2_ps_spec/"...
                "TimeTable3_ps_spec/"...
                "TimeTable4_ps_spec/"...
                "TimeTable5_ps_spec/"...
                "TimeTable6_ps_spec/"...
                "TimeTable7_ps_spec/"...
                "TimeTable8_ps_spec/"...
```

```
                 "TimeTable9_ps_spec/"...
                 "TimeTable10_ps_spec/"...
                 "TimeTable11_ps_spec/"
               ];

% Time domain features produced by Matlab DFD
featureNames_Freq = [
                 "BandPower"... %1
                 "PeakAmp1"... %2
                 "PeakFreq1"... %3
                ];
numOfFeatures_Freq = size(featureNames_Freq, 2);

%% Load features from .mat files
load("DFD_FeatureTable_TimeDomain_Standard.mat");
load("DFD_FeatureTable_FreqDomain_Standard.mat");

load("DFD_FeatureTable_TimeDomain_DcRemoved.mat");
load("DFD_FeatureTable_FreqDomain_DcRemoved.mat");

load("DFD_FeatureTable_TimeDomain_Padded.mat");
load("DFD_FeatureTable_FreqDomain_Padded.mat");

load("DFD_FeatureTable_TimeDomain_DcRemovedPadded.mat");
load("DFD_FeatureTable_FreqDomain_DcRemovedPadded.mat");

%% Convert DFD output from Timetables to array
% Combine time domain features with frequency domain features
% Output is an array numOfPulses x numOfSignals x numOfFeatures
numOfTimeFeatures = size(featureNames_Time, 2);
numOfFreqFeatures = size(featureNames_Freq, 2);
numOfPulses = size(pulseData_Timetable_Standard, 1);
totalFeatures = numOfTimeFeatures + numOfFreqFeatures;
featureArray_Standard = zeros(numOfPulses, totalFeatures, numOfSignals);
featureArray_DcRemoved = zeros(numOfPulses, totalFeatures, numOfSignals);
featureArray_Padded = zeros(numOfPulses, totalFeatures, numOfSignals);
featureArray_DcRemovedPadded = zeros(numOfPulses, totalFeatures,
    numOfSignals);

%% Take data from the time and freq domain features and combine them into an
% array
for signalIndex = 1 : numOfSignals
    for featureIndex = 1 : numOfTimeFeatures
        featureArray_Standard(:, featureIndex, signalIndex) =
            DFD_FeatureTable_TimeDomain_Standard.(signalNames_Time(
            signalIndex) + featureNames_Time(featureIndex));
        featureArray_DcRemoved(:, featureIndex, signalIndex) =
            DFD_FeatureTable_TimeDomain_DcRemoved.(signalNames_Time(
            signalIndex) + featureNames_Time(featureIndex));
        featureArray_Padded(:, featureIndex, signalIndex) =
            DFD_FeatureTable_TimeDomain_Padded.(signalNames_Time(signalIndex
            ) + featureNames_Time(featureIndex));
        featureArray_DcRemovedPadded(:, featureIndex, signalIndex) =
            DFD_FeatureTable_TimeDomain_DcRemovedPadded.(signalNames_Time(
            signalIndex) + featureNames_Time(featureIndex));
    end
    for featureIndex = 1 : numOfFreqFeatures
        featureArray_Standard(:, featureIndex + numOfTimeFeatures,
            signalIndex) = DFD_FeatureTable_FreqDomain_Standard.(
            signalNames_Freq(signalIndex) + featureNames_Freq(featureIndex))
            ;
        featureArray_DcRemoved(:, featureIndex + numOfTimeFeatures,
            signalIndex) = DFD_FeatureTable_FreqDomain_DcRemoved.(
            signalNames_Freq(signalIndex) + featureNames_Freq(featureIndex))
            ;
        featureArray_Padded(:, featureIndex + numOfTimeFeatures, signalIndex
            )  = DFD_FeatureTable_FreqDomain_Padded.(signalNames_Freq(
            signalIndex) + featureNames_Freq(featureIndex));
        featureArray_DcRemovedPadded(:, featureIndex + numOfTimeFeatures,
            signalIndex)  = DFD_FeatureTable_FreqDomain_DcRemovedPadded.(
            signalNames_Freq(signalIndex) + featureNames_Freq(featureIndex))
            ;
    end
end

FeatureNames_Combined = [featureNames_Time featureNames_Freq];
numOfFeatures_Combined = size(FeatureNames_Combined,2);
```

```
function [] = fnPlotSignals(rawDataArray, fileNum, signalIndex1,
    signalIndex2, sensorNames)
%{
    Date: 2023/08/23
    Filename: fnPlotSignals.m
    Author: Paul Barron
    Description: This function plots one signal against another signal
%}
    figure();
    numpoints = size(cell2mat(rawDataArray{fileNum}(1, 2)), 1);
    pointidx = 1 : numpoints;
    sig1 = cell2mat(rawDataArray{fileNum}(signalIndex1, 2));
    sig2 = cell2mat(rawDataArray{fileNum}(signalIndex2, 2));
    scatter(sig1, sig2, 3, pointidx, 'filled');
    colormap jet
    colorbar
    xlabel(char(sensorNames(signalIndex1)));
    ylabel(char(sensorNames(signalIndex2)));
    filename = "..\Latex Document\figures\File" + fileNum ...
        + "_Signal" + signalIndex1 ...
        + "vSignal" + signalIndex2;
    print(filename,'-depsc','-tiff');
    filename = "..\Presentation\File" + fileNum ...
        + "_Signal" + signalIndex1 ...
        + "vSignal" + signalIndex2;
    export_fig(filename, '-svg');
end
```

```
function [] = fnPlotCorrelations(rawDataArray, r2Values, fileNum, lowerBound
    , upperBound, signalName, sensorNames)
%{
    Date: 2023/08/23
    Filename: fnPlotCorrelations.m
    Author: Paul Barron
    Description: This function plots the signals against each other as well
    as printing the R2 values on the plots which are within the specified
    range.
%}
```

```matlab
figure();
    tiledlayout(12,12,'TileSpacing','None', 'Padding','tight');
    numOfSignals = size(signalName, 2);
    numpoints = size(cell2mat(rawDataArray{fileNum}(1, 2)), 1);
    pointidx = 1 : numpoints;
    for signalIndex1 = 1:numOfSignals
        for signalIndex2 = signalIndex1:numOfSignals
            if signalIndex1 ~= signalIndex2
                sig1 = cell2mat(rawDataArray{fileNum}(signalIndex1, 2));
                sig2 = cell2mat(rawDataArray{fileNum}(signalIndex2, 2));
                nexttile((signalIndex1-1) * numOfSignals + signalIndex2);
                scatter(sig1, sig2, 2, pointidx, 'filled');
                colormap jet;
                xticklabels({});
                yticklabels({});
                DataX = interp1( [0 1], xlim(), 0.5 );
                DataY = interp1( [0 1], ylim(), 0.5 );
                r2Val = r2Values(fileNum, signalIndex1, signalIndex2);
                if r2Val > lowerBound && r2Val < upperBound
                    text(DataX, DataY, num2str(r2Val, 2), '
                        HorizontalAlignment','center');
                end
            end
        end
    end
    % Add labels for signal names on the diagonals
    for i = 1:size(sensorNames, 2)
        nexttile((i-1) * numOfSignals + i);
        str = char(sensorNames(i));
        text(0.5, 0.5, str(1:5), HorizontalAlignment='center',
            VerticalAlignment='middle');
        xticklabels({});
        yticklabels({});
    end
end
```

```matlab
%{
    Date: 2023/08/23
    Filename: SignalCorrelation.mlx
    Author: Paul Barron
    Description: This script calculates correlations of the raw original
    signals and then plots all the combinations of signals within that
    file.
%}
%%
% Calculat the R2 values for each combination of signals in each file
warning('off','all')

rawDataArray = {numOfFiles};
for fileIndex = 1 : numOfFiles
    % Load the data for the current file
    rawDataStruct = load(dataSubfolder + filenames(fileIndex) + ".mat");
    rawDataArray{fileIndex} = struct2array(rawDataStruct);
end

r2Val = zeros(numOfSignals, numOfSignals);
r2Values = zeros(numOfFiles, numOfSignals, numOfSignals);
for fileIndex = 1 : numOfFiles
```

```
    for sensorIndex1 = 1 : numOfSignals
        for sensorIndex2 = 1 : numOfSignals
            sig1 = rawDataArray{fileIndex}{sensorIndex1,2};
            sig2 = rawDataArray{fileIndex}{sensorIndex2,2};
            mdl = fitlm(sig1, sig2);
            R2 = mdl.Rsquared.Ordinary;
            r2Val(sensorIndex1, sensorIndex2) = R2;
            r2Values(fileIndex, sensorIndex1, sensorIndex2) = R2;
        end
    end
end
%%
% Print one of the matrices to Latex file
addpath('C:\Users\PaulBarron\Dropbox\Thesis\matrix2latex')
fileNum = 4;
M = r2Values(fileNum, :, 1)';
for i = 2:size(r2Values,3)
    M = [ M r2Values(fileNum,:, i)' ];
end

rowLabels = { '07', '10', '12', '17', '20', '28', '31', '32', '33', '34', '
    35', '36' };
columnLabels = rowLabels;

matrix2latex(M, "..\Latex Document\tables\correlationTable" + fileNum + ".
    tex", 'rowLabels', rowLabels, 'columnLabels', columnLabels, 'alignment',
     'c', 'format', '%-6.2f', 'size', 'tiny');
%% Plot table for the correlation of each raw signal in a particular file

arr = [1:1:12];
labels = arrayfun(@num2str, arr, 'UniformOutput', 0);
for fileIndex = 1:numOfFiles
    fprintf("Filenumber #%i", fileIndex)
    arrTable = array2table(squeeze(r2Values(fileIndex, :, :)), '
        VariableNames', labels)
end
%% File 3

fileNum = 3;
arrTable = array2table(squeeze(r2Values(fileNum, :, :)), 'VariableNames',
    labels)
%%
%fnPlotCorrelations(rawDataArray, r2Values, fileNum, 0.2, 0.90, sensorNames,
     sensorNames);
filename = '..\Latex Document\figures\RawSignalCorrelationsFile' + fileNum;
%print(filename,'-depsc','-tiff');
%%
figure();
fileNum = 3;
signalIndex1 = 3;
signalIndex2 = 6;

numpoints = size(cell2mat(rawDataArray{fileNum}(1, 2)), 1);
pointidx = 1 : numpoints;
sig1 = cell2mat(rawDataArray{fileNum}(signalIndex1, 2));
sig2 = cell2mat(rawDataArray{fileNum}(signalIndex2, 2));
scatter(sig1, sig2, 3, pointidx, 'filled');
colormap jet
```

```matlab
colorbar
xlabel(char(sensorNames(signalIndex1)));
ylabel(char(sensorNames(signalIndex2)));
filename = "..\Latex Document\figures\File" + fileNum ...
    + "_Signal" + signalIndex1 ...
    + "vSignal" + signalIndex2;
print(filename,'-depsc','-tiff');
export_fig(filename, '-svg');
%%
figure();
fileNum = 3;
signalIndex1 = 5;
signalIndex2 = 6;

numpoints = size(cell2mat(rawDataArray{fileNum}(1, 2)), 1);
pointidx = 1 : numpoints;
sig1 = cell2mat(rawDataArray{fileNum}(signalIndex1, 2));
sig2 = cell2mat(rawDataArray{fileNum}(signalIndex2, 2));
scatter(sig1, sig2, 3, pointidx, 'filled');
colormap jet
colorbar
xlabel(char(sensorNames(signalIndex1)));
ylabel(char(sensorNames(signalIndex2)));
filename = "..\Latex Document\figures\File" + fileNum ...
    + "_Signal" + signalIndex1 ...
    + "vSignal" + signalIndex2;
print(filename,'-depsc','-tiff');
export_fig(filename, '-svg');
%%
figure();
fileNum = 3;
signalIndex1 = 7;
signalIndex2 = 8;

numpoints = size(cell2mat(rawDataArray{fileNum}(1, 2)), 1);
pointidx = 1 : numpoints;
sig1 = cell2mat(rawDataArray{fileNum}(signalIndex1, 2));
sig2 = cell2mat(rawDataArray{fileNum}(signalIndex2, 2));
scatter(sig1, sig2, 3, pointidx, 'filled');
colormap jet
colorbar
xlabel(char(sensorNames(signalIndex1)));
ylabel(char(sensorNames(signalIndex2)));
filename = "..\Latex Document\figures\File" + fileNum ...
    + "_Signal" + signalIndex1 ...
    + "vSignal" + signalIndex2;
%print(filename,'-depsc','-tiff');
%% File 4

fileNum = 4;
arrTable = array2table(squeeze(r2Values(fileNum, :, :)), 'VariableNames',
    labels)
%%
%fnPlotCorrelations(rawDataArray, r2Values, fileNum, 0.2, 0.90, sensorNames,
     sensorNames);
filename = "..\Latex Document\figures\RawSignalCorrelationsFile" + fileNum;
%print(filename,'-depsc','-tiff');
%%
```

```matlab
fnPlotSignals(rawDataArray, fileNum, 3, 6, sensorNames);
%%
fnPlotSignals(rawDataArray, fileNum, 1, 7, sensorNames);
%%
fnPlotSignals(rawDataArray, fileNum, 1, 8, sensorNames);
%%
fnPlotSignals(rawDataArray, fileNum, 1, 9, sensorNames);
%% File 5

fileNum = 5;
arrTable = array2table(squeeze(r2Values(fileNum, :, :)), 'VariableNames',
    labels)
%%
%fnPlotCorrelations(rawDataArray, r2Values, fileNum, 0.2, 0.90, sensorNames,
     sensorNames);
filename = '..\Latex Document\figures\RawSignalCorrelationsFile' + fileNum;
%print(filename,'-depsc','-tiff');
%%
fnPlotSignals(rawDataArray, fileNum, 3, 6, sensorNames);
%% File 8

fileNum = 8;
arrTable = array2table(squeeze(r2Values(fileNum, :, :)), 'VariableNames',
    labels)
%%
%fnPlotCorrelations(rawDataArray, r2Values, fileNum, 0.2, 0.90, sensorNames,
     sensorNames);
filename = '..\Latex Document\figures\RawSignalCorrelationsFile' + fileNum;
%print(filename,'-depsc','-tiff');
%%
fnPlotSignals(rawDataArray, fileNum, 3, 6, sensorNames);
%% File 13

fileNum = 13;
arrTable = array2table(squeeze(r2Values(fileNum, :, :)), 'VariableNames',
    labels)
%%
%fnPlotCorrelations(rawDataArray, r2Values, fileNum, 0.2, 0.90, sensorNames,
     sensorNames);
filename = '..\Latex Document\figures\RawSignalCorrelationsFile' + fileNum;
%print(filename,'-depsc','-tiff');
%%
fnPlotSignals(rawDataArray, fileNum, 3, 6, sensorNames);
%%
[FullPath,Filename,ext]=fileparts(matlab.desktop.editor.getActiveFilename);
currentFile = strcat(Filename, ext);
path = export(currentFile, Format="m", OpenExportedFile=false);
```

```matlab
function [R2] = fnPlotFeatureVsFeature_Array(featureArray, lowerBound,
    upperBound, signalIndex, featureNames, sensorNames)
%{
    Date: 2023/08/23
    Filename: fnPlotFeatureVsFeature.m
    Author: Paul Barron
    Description: This function plots all the features against each other
    and prints the R2 values on the charts which are within the specified
    range.
%}
```

```
    figure();
    numpoints = size(featureArray, 1);
    numOfFeatures = size(featureArray, 2);
    tiledlayout(numOfFeatures, numOfFeatures, 'TileSpacing','None', 'Padding
        ','tight');
    fprintf("Signal %i: %s", signalIndex, sensorNames(signalIndex));
    R2 = zeros(numOfFeatures, numOfFeatures);
    pointidx = 1 : numpoints;
    % Loop through features #1
    for featureIndex1 = 1:numOfFeatures
        % Loop through features #2
        for featureIndex2 = featureIndex1+1:numOfFeatures
            sig1 = featureArray(:, featureIndex1, signalIndex);
            sig2 = featureArray(:, featureIndex2, signalIndex);

            mdl = fitlm(sig1, sig2);
            R2(featureIndex1, featureIndex2) = mdl.Rsquared.Ordinary;

            nexttile((featureIndex1-1) * numOfFeatures + featureIndex2);
            scatter(sig1, sig2, 3, pointidx, 'filled');
            colormap( jet(numpoints) );
            xticklabels({});
            yticklabels({});

            DataX = interp1( [0 1], xlim(), 0.5 );
            DataY = interp1( [0 1], ylim(), 0.5 );

            if R2(featureIndex1, featureIndex2) > lowerBound && R2(
                featureIndex1, featureIndex2) < upperBound
                text(DataX, DataY, num2str(R2(featureIndex1, featureIndex2),
                    2), 'HorizontalAlignment','center');
            end
        end
    end
    for featureIndex = 1 : numOfFeatures
        nexttile((featureIndex-1) * numOfFeatures + featureIndex);
        text(0.5, 0.5, int2str(featureIndex), HorizontalAlignment='center',
            VerticalAlignment='middle');
        xticklabels({});
        yticklabels({});
        R2(featureIndex, featureIndex) = 1;
    end
end
```

```
function [R2] = fnPlotSignalVsSignal(FeatureTable, lowerBound, upperBound,
    featureIndex, featureNames, signalName, sensorNames)
%{
    Date: 2023/08/23
    Filename: fnPlotSignalVsSignal.m
    Author: Paul Barron
    Description: This function plots each signal against every other signal
    for a particular feature
%}
    figure();
    tiledlayout(12,12,'TileSpacing','None', 'Padding','tight');
    numpoints = size(FeatureTable.(signalName(1) + featureNames(1)), 1);
    pointidx = 1 : numpoints;
    numOfSignals = size(signalName, 2);
    R2 = zeros(numOfSignals, numOfSignals);
    for signalIndex1 = 1:numOfSignals
        for signalIndex2 = signalIndex1:numOfSignals
            if signalIndex1 ~= signalIndex2
                sig1 = FeatureTable.(signalName(signalIndex1) + featureNames
                    (featureIndex));
                sig2 = FeatureTable.(signalName(signalIndex2) + featureNames
                    (featureIndex));
                nexttile((signalIndex1-1) * numOfSignals + signalIndex2);
                scatter(sig1, sig2, 3, pointidx, 'filled');
                colormap( jet(numpoints) );
                xticklabels({});
                yticklabels({});

                mdl = fitlm(sig1, sig2);
                R2(signalIndex1, signalIndex2) = mdl.Rsquared.Ordinary;

                DataX = interp1( [0 1], xlim(), 0.5 );
                DataY = interp1( [0 1], ylim(), 0.5 );
                if R2(signalIndex1, signalIndex2) > lowerBound && R2(
                    signalIndex1, signalIndex2) < upperBound
                    text(DataX, DataY, num2str(R2(signalIndex1, signalIndex2
                        ), 2), 'HorizontalAlignment','center');
                end
            end
        end
    end
    % Add labels for signal names on the diagonals
    for i = 1:size(sensorNames, 2)
        nexttile((i-1) * numOfSignals + i);
        str = char(sensorNames(i));
        text(0.5, 0.5, str(1:5), HorizontalAlignment='center',
            VerticalAlignment='middle');
        xticklabels({});
        yticklabels({});
        R2(signalIndex1, signalIndex1) = 1;
    end
end
```

```
%{
    Date: 2023/08/23
    Filename: FeatureVsFeature_Combined.mlx
    Author: Paul Barron
    Description:
%}
%%
saveFigures = true;
FeatureNames = FeatureNames_Combined;
for signalIndex = 1 : numOfSignals
    R2 = fnPlotFeatureVsFeature_Array(featureArray_Standard, 0.3, 0.8,
        signalIndex, FeatureNames, sensorNames);
    filename = "..\Latex Document\figures\FeatureVsFeatureCombinedSignal" +
        signalIndex;
    if (saveFigures == true)
        print(filename,'-depsc','-tiff');
    end
end
%%
```

```
[FullPath,Filename,ext]=fileparts(matlab.desktop.editor.getActiveFilename);
currentFile = strcat(Filename, ext);
path = export(currentFile, Format="m", OpenExportedFile=false);
```

```
%{
    Date: 2023/08/23
    Filename: SignalVsSignal.mlx
    Author: Paul Barron
    Description:
%}
%%
warning('off','all');
Features = DFD_FeatureTable_TimeDomain_Standard;
saveFigures = true;
for featureIndex = 1:numOfFeatures_Time
    fprintf("Feature %i: %s", featureIndex, FeatureNames_Combined(
        featureIndex));
    fnPlotSignalVsSignal(Features, 0.2, 0.9, featureIndex, featureNames_Time
        , signalNames_Time, sensorNames);
    filename = "..\Latex Document\figures\SignalVsSignalFeatureTime" +
        featureIndex;
    if saveFigures
        %export_fig(filename, '-eps', '-depsc');
        print(filename,'-depsc','-tiff');
    end

    %fnPlotSignalVsSignal(DFD_FeatureTable_TimeDomain_DcRemoved, 0.2, 0.9,
        featureIndex, FeatureNames, signalName, sensorNames);
    %fnPlotSignalVsSignal(DFD_FeatureTable_TimeDomain_Padded, 0.2, 0.9,
        featureIndex, FeatureNames, signalNames, sensorNames);
    %fnPlotSignalVsSignal(DFD_FeatureTable_TimeDomain_DcRemovedPadded, 0.2,
        0.9, FeatureIndex, featureNames, signalNames, sensorNames);
end
%%
Features = DFD_FeatureTable_FreqDomain_Standard;
for featureIndex = 1:numOfFeatures_Freq
    featureIndex
    fnPlotSignalVsSignal(Features, 0.2, 0.9, featureIndex, featureNames_Freq
        , signalNames_Freq, sensorNames);
    filename = "..\Latex Document\figures\SignalVsSignalFeatureFreq" +
        featureIndex;
    if saveFigures
        %export_fig(filename, '-eps', '-depsc');
        print(filename,'-depsc','-tiff');
    end

    %fnPlotSignalVsSignal(DFD_FeatureTable_TimeDomain_DcRemoved, 0.2, 0.9,
        featureIndex, FeatureNames, signalName, sensorNames);
    %fnPlotSignalVsSignal(DFD_FeatureTable_TimeDomain_Padded, 0.2, 0.9,
        featureIndex, FeatureNames, signalNames, sensorNames);
    %fnPlotSignalVsSignal(DFD_FeatureTable_TimeDomain_DcRemovedPadded, 0.2,
        0.9, FeatureIndex, featureNames, signalNames, sensorNames);
end
%%
[FullPath,Filename,ext]=fileparts(matlab.desktop.editor.getActiveFilename);
currentFile = strcat(Filename, ext);
path = export(currentFile, Format="m", OpenExportedFile=false);
```

```matlab
function [beta] = fnPlotModel(Y, X1, txtTitle, labelY, labelX1)
%{
    Date: 2023/08/23
    Filename: SignalVsSignal.mlx
    Author: Paul Barron
    Description: This function calculates a linear model using least
    squares and calculates the normalized FIT value. It plot the data using
    a scatter plot and then plots the linear model over the top of the
    data.
%}
    n = size(Y, 1);
    X = [ones(n,1) X1];
    k = size(X, 2) - 1; % Number of parameters for model
    beta = inv(X' * X) * (X' * Y);

    Y_est = X * beta;
    RSS = sum((Y_est - Y).^2);
    RSS = var(Y_est - Y) / var(Y);

    figure(); hold on;
    pointidx = 1 : n;
    scatter(X1, Y, 10, pointidx, 'o', 'filled');
    colormap( jet(n) );
    interval = max(X1) - min(X1);
    x1fit = min(X1):interval:max(X1);
    yfit = beta(1) + beta(2).*x1fit;
    plot(x1fit, yfit);
    xlabel(labelX1, 'Interpreter', 'none');
    ylabel(labelY, 'Interpreter', 'none');

    % Fit equation from compare function
    FitValue = 100 * (1-norm(Y-Y_est)/norm(Y-mean(Y)));
    stringValue = sprintf('%.1f', FitValue);
    annotation('textbox', [0.2 0.8, 0.1, 0.1], ...
        'String', convertStringsToChars(stringValue), ...
        HorizontalAlignment= 'center', ...
        VerticalAlignment='middle', ...
        FitBoxToText='on');
end
```

```matlab
function [Y_est] = fnModelPlot3D(Y, X1, X2, txtTitle, labelY, labelX1,
    labelX2)
%{
    Date: 2023/08/23
    Filename: SignalVsSignal.mlx
    Author: Paul Barron
    Description: This function calculates a 3D model plus the normalized
    FIT value. It then plots the original data, the model and the FIT
    value.
%}
    n = size(Y, 1);
    X = [ones(n,1) X1 X2];
    beta = inv(X' * X) * (X' * Y);

    Y_est = X * beta;
    %RSS = sum((Y_est - Y).^2);
    %RSS = var(Y_est - Y) / var(Y);

    figure();
    scatter3(X1,X2,Y,'filled');
    hold on;

    interval1 = (max(X1) - min(X1))/20;
    x1fit = min(X1):interval1:max(X1);
    interval2 = (max(X2) - min(X2))/20;
    x2fit = min(X2):interval2:max(X2);
    [X1FIT,X2FIT] = meshgrid(x1fit, x2fit);

    YFIT = beta(1) + beta(2).*X1FIT + beta(3).*X2FIT;
    %YFIT = b(1) + b(2)*X1FIT + b(3)*X2FIT + b(4)*X1FIT.*X2FIT;
    mesh(X1FIT, X2FIT, YFIT);

    xlabel(labelX1, 'Interpreter', 'none');
    ylabel(labelX2, 'Interpreter', 'none');
    zlabel(labelY, 'Interpreter', 'none');
    fprintf("%s", txtTitle);

    % Fit equation from compare function
    FitValue = 100 * (1-norm(Y-Y_est)/norm(Y-mean(Y)));
    stringValue = sprintf('%.1f', FitValue);
    annotation('textbox', [0.1 0.8, 0.1, 0.1], ...
        'String', convertStringsToChars(stringValue),...
        HorizontalAlignment= 'center', ...
        VerticalAlignment='middle');
end
```

```
%{
    Date: 2023/08/23
    Filename: MultivariateModel_FeatureVsFeatureArray_Signal3.mlx
    Author: Paul Barron
    Description: This script calls the function that plots the signal
    features against each other for every signal
%}
%%
lowerR2 = 0.3;
upperR2 = 0.8;
r2ReducedThreshold = 0.9;
selectedSignal = 3; %21:12 Actual moment over Rolls (Nm)
saveFigures = true;
R2 = fnPlotFeatureVsFeature_Array(featureArray_Standard, lowerR2, upperR2,
    selectedSignal, FeatureNames_Combined, sensorNames);
%%
disp(num2str(R2, '%.2f  '));
%% Combination 1

selectedFeature = 1;
[selectedFeatureArray, selectedFeatureData, numOfSelectedFeatures] =
    fnSelectFeatures( ...
    selectedSignal, selectedFeature, featureArray_Standard,R2, upperR2,
        lowerR2, r2ReducedThreshold, true);
%%
fig = figure(); hold on;
sensorNames(selectedSignal)
xlabel("Pulses");
ylabel("Feature value");
leg = {numOfSelectedFeatures};
```

```matlab
secondaryAxisArrayIndex = [2 3];
legendAxis = [];
for i = 1:numOfSelectedFeatures
    if ismember(i, secondaryAxisArrayIndex)
        yyaxis right;
        ha = plot(selectedFeatureData{i});
        legendAxis = [legendAxis ha];
    else
        yyaxis left;
        hb = plot(selectedFeatureData{i});
        legendAxis = [legendAxis hb];
    end
    leg{i} = char(FeatureNames_Combined(selectedFeatureArray(i)));
end
hold off;
lgnd = legend(legendAxis, leg);
set(lgnd,'color','none');
if (saveFigures == true)
    filename = "..\Latex Document\figures\Models_Signal" + selectedSignal +
        "Feature" + selectedFeature;
    print(filename,'-depsc','-tiff')
    %export_fig(filename, '-eps', '-depsc');
end
%%
figure(); hold on;
x1 = selectedFeatureData{1};
x2 = selectedFeatureData{2};
x3 = selectedFeatureData{3};
t = 1:numOfPulses;
hl1 = line(t,x1,'Color','r');
ax1 = gca;
set(ax1,'XColor','r','YColor','r')
ax2 = axes('XAxisLocation','top',...
          'YAxisLocation','right',...
          'Color','none',...
          'XColor','k','YColor','k');
hl2 = line(t,x2,'Parent', ax2, 'Color','k');
hl3 = line(t,x3,'Parent', ax2, 'Color','k');
%%
xx = (1:numOfPulses)';
plotyyy(xx, selectedFeatureData{1}, xx, selectedFeatureData{2}, xx,
    selectedFeatureData{3}, leg);
sensorNames(selectedSignal)
xlabel("Pulses");
%%
selectedFeatureIdx = find(selectedFeatureArray == selectedFeature);
linearModelNum = 1;
for index = 1 : numOfSelectedFeatures
    if selectedFeatureArray(index) ~= selectedFeature
        fnPlotModel(selectedFeatureData{index}, ...
            selectedFeatureData{selectedFeatureIdx}, ...
            sensorNames(selectedSignal), ...
            FeatureNames_Combined(selectedFeatureArray(index)), ...
            FeatureNames_Combined(selectedFeature));
        if (saveFigures == true)
            filename = "..\Latex Document\figures\Models_Signal" +
                selectedSignal ...
                + "Feature" + selectedFeature ...
                + "_Linear" + linearModelNum;
            export_fig(filename, '-eps', '-depsc');
            linearModelNum = linearModelNum + 1;
        end
    end
end
%%
multivariateModelNum = 1;
if numOfSelectedFeatures > 2
    for index1 = 1 : numOfSelectedFeatures
        for index2 = index1+1 : numOfSelectedFeatures
            if selectedFeatureArray(index1) ~= selectedFeature &&
                selectedFeatureArray(index2) ~= selectedFeature
                Y_est = fnPlotModel3D( ...
                    selectedFeatureData{selectedFeatureIdx}, ...
                    selectedFeatureData{index1}, ...
                    selectedFeatureData{index2}, ...
                    sensorNames(selectedSignal), ...
                    FeatureNames_Combined(selectedFeatureArray(
                        selectedFeatureIdx)), ...
                    FeatureNames_Combined(selectedFeatureArray(index1)), ...
                    FeatureNames_Combined(selectedFeatureArray(index2)));
                if (saveFigures == true)
                    filename = "..\Latex Document\figures\Models_Signal" +
                        selectedSignal ...
                        + "Feature" + selectedFeature ...
                        + "_Multivariate" + multivariateModelNum;
                    export_fig(filename, '-eps', '-depsc');
                    multivariateModelNum = multivariateModelNum + 1;
                end
            end
        end
    end
end
%%
[FullPath,Filename,ext]=fileparts(matlab.desktop.editor.getActiveFilename);
currentFile = strcat(Filename, ext);
path = export(currentFile, Format="m", OpenExportedFile=false);
```

```
%{
    Date: 2023/08/23
    Filename: MultivariateModel_SignalVsSignal_TimeFeature2.mlx
    Author: Paul Barron
    Description: This script calls the function that plots the signal
    features against each other for every feature
%}
%%
Features = DFD_FeatureTable_TimeDomain_Standard;
SignalNames = signalNames_Time;
FeatureNames = featureNames_Time;
lowerR2 = 0.3;
upperR2 = 0.8;
r2ReducedThreshold = 0.9;
selectedFeature = 2; % Crest Factor
saveFigures = true;
R2 = fnPlotSignalVsSignal(Features, lowerR2, upperR2, selectedFeature,
    FeatureNames, SignalNames, sensorNames);
%%
```

```
disp(num2str(R2, '%.2f  '));
%% Combination 1

selectedSignal = 1;
[selectedSignalArray, selectedSignalData, numOfSelectedSignals] =
    fnSelectSignals(...
    selectedSignal, selectedFeature, featureArray_Standard, R2, upperR2,
        lowerR2, r2ReducedThreshold, true);
%%
figure(); hold on;
FeatureNames(selectedFeature);%title(FeatureNames(selectedFeature));
xlabel("Pulses");
ylabel(FeatureNames_Combined(selectedFeature));
secondaryAxisArrayIndex = [2];
for i = 1:numOfSelectedSignals
    if ismember(i, secondaryAxisArrayIndex)
        yyaxis right;
    else
        yyaxis left;
    end
    plot(selectedSignalData{i});
    leg{i} = char(sensorNames(selectedSignalArray(i)));
end
legend(leg);
if (saveFigures == true)
    filename = "..\Latex Document\figures\Models_Feature" + selectedFeature
        ...
        + "Signal" + selectedSignal;
    %print(filename,'-depsc','-tiff');
    export_fig(filename, '-depsc');
end
%%
selectedSignalIdx = find(selectedSignalArray == selectedSignal);
linearModelNum = 1;
for index = 1 : numOfSelectedSignals
    if selectedSignalArray(index) ~= selectedSignal
        fnPlotModel(selectedSignalData{selectedSignalIdx}, ...
            selectedSignalData{index}, ...
            FeatureNames_Combined(selectedFeature), ...
            sensorNames(selectedSignalArray(index)), ...
            sensorNames(selectedSignal));
        if (saveFigures == true)
            filename = "..\Latex Document\figures\Models_Feature" +
                selectedFeature ...
                + "Signal" + selectedSignal ...
                + "_Linear" + linearModelNum;
            %export_fig(filename, '-eps', '-depsc');
            print(filename, '-depsc');
            linearModelNum = linearModelNum + 1;
        end
    end
end
%%
multivariateModelNum = 1;
if numOfSelectedSignals > 2
    for index1 = 1 : numOfSelectedSignals
        for index2 = index1+1 : numOfSelectedSignals
            if selectedSignalArray(index1) ~= selectedSignal &&
                selectedSignalArray(index2) ~= selectedSignal
                Y_est = fnPlotModel3D( ...
                    selectedSignalData{selectedSignalIdx}, ...
                    selectedSignalData{index1}, ...
                    selectedSignalData{index2}, ...
                    FeatureNames(selectedFeature), ...
                    sensorNames(selectedSignalArray(selectedSignalIdx)), ...
                    sensorNames(selectedSignalArray(index1)), ...
                    sensorNames(selectedSignalArray(index2)));

                if (saveFigures == true)
                    filename = "..\Latex Document\figures\Models_Feature" +
                        selectedFeature ...
                        + "Signal" + selectedSignal ...
                        + "_Multivariate" + multivariateModelNum;
                    export_fig(filename, '-eps', '-depsc');
                    multivariateModelNum = multivariateModelNum + 1;
                end
            end
        end
    end
end
%%
[FullPath,Filename,ext]=fileparts(matlab.desktop.editor.getActiveFilename);
currentFile = strcat(Filename, ext);
path = export(currentFile, Format="m", OpenExportedFile=false);
```

Appendix B Feature vs Feature Plots

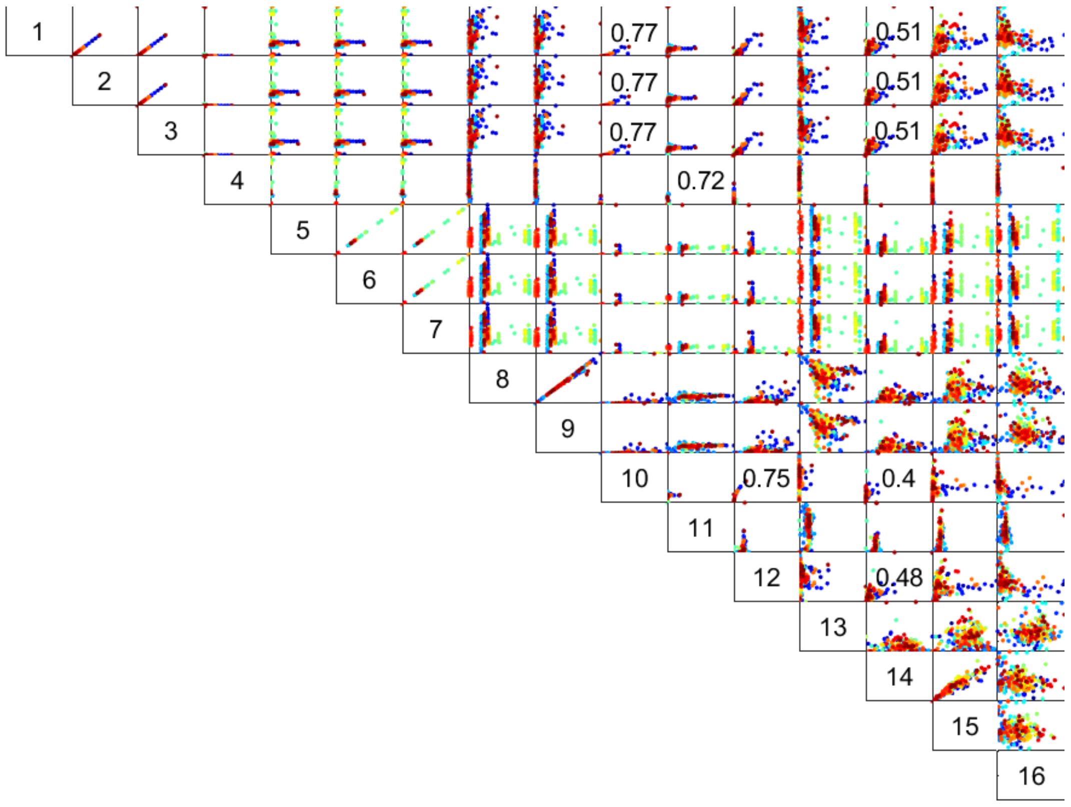


Figure B.1. Feature vs Feature for Signal 1

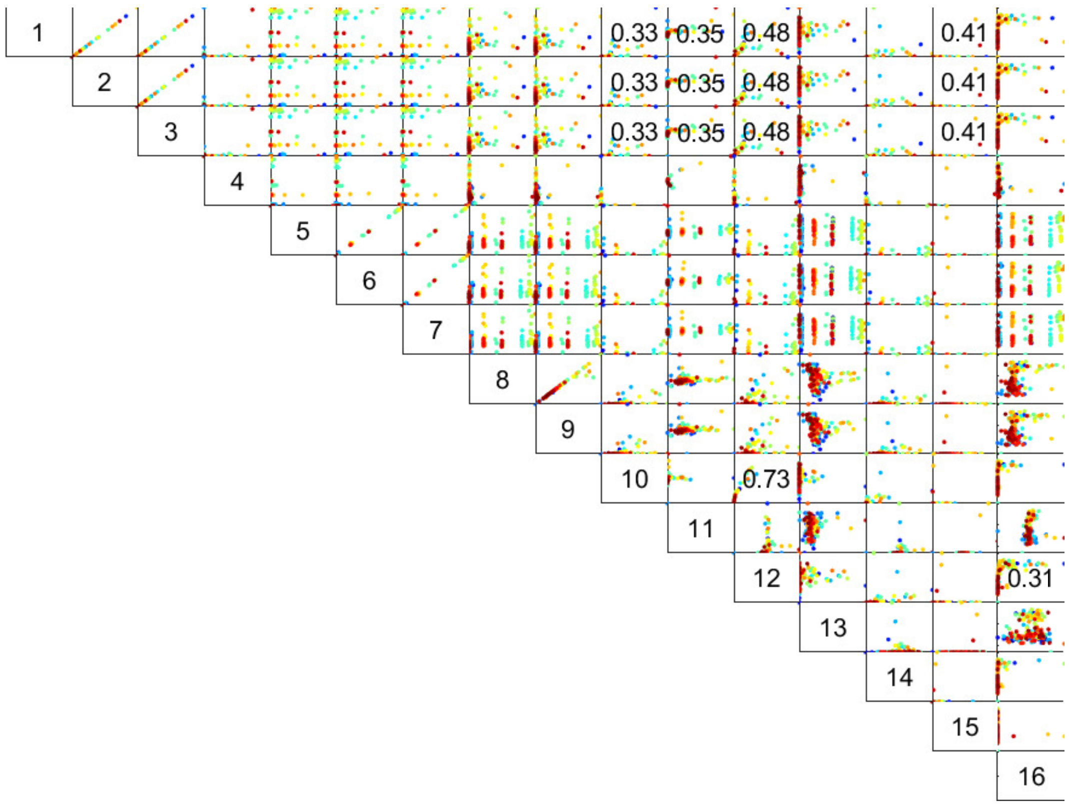


Figure B.2. Feature vs Feature for Signal 2

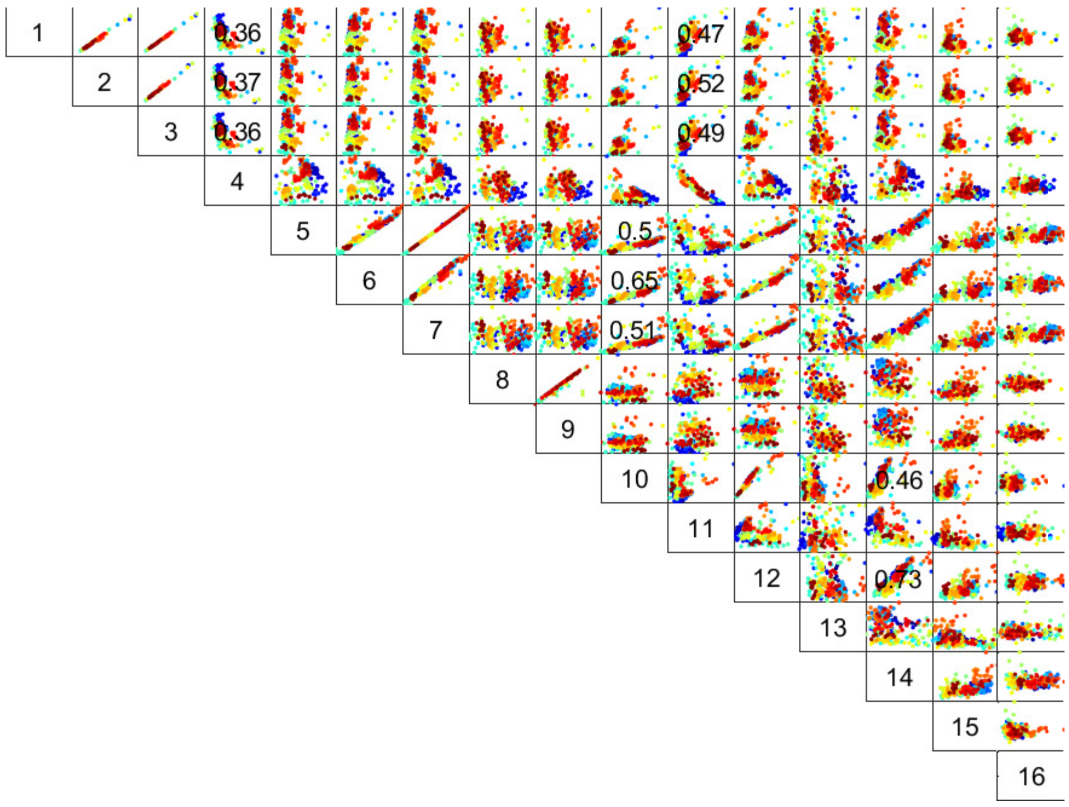


Figure B.3. Feature vs Feature for Signal 3

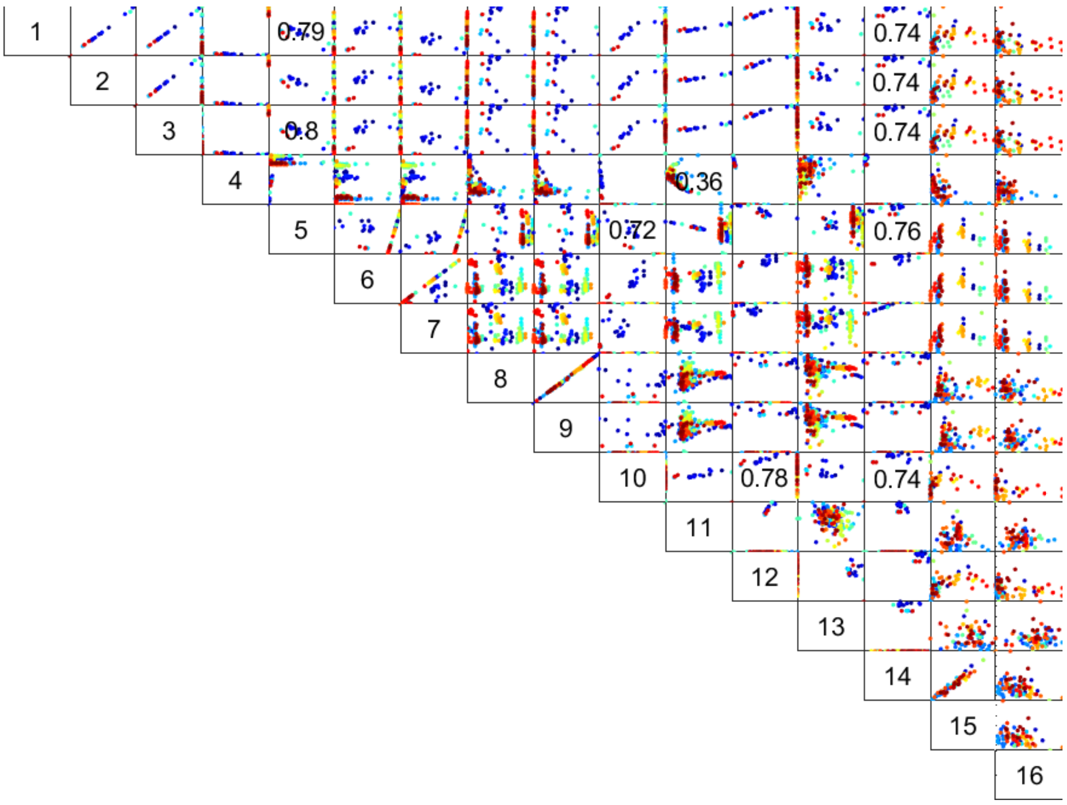


Figure B.4. Feature vs Feature for Signal 4

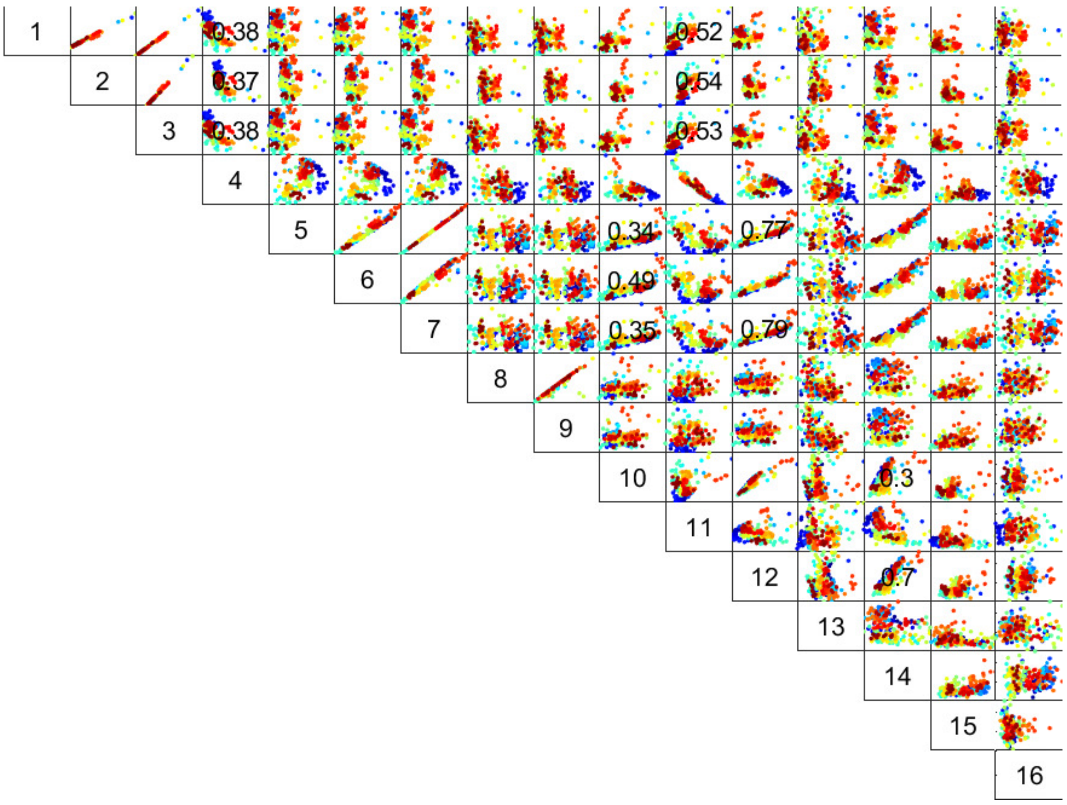


Figure B.5. Feature vs Feature for Signal 5

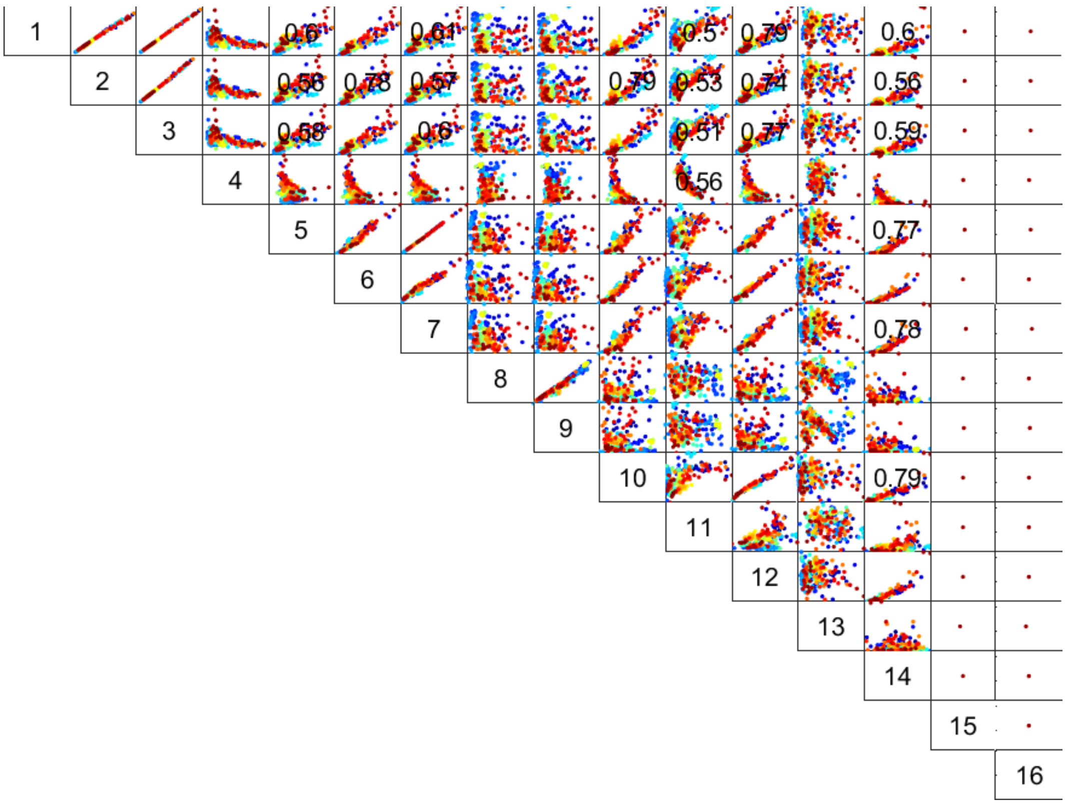


Figure B.6. Feature vs Feature for Signal 6

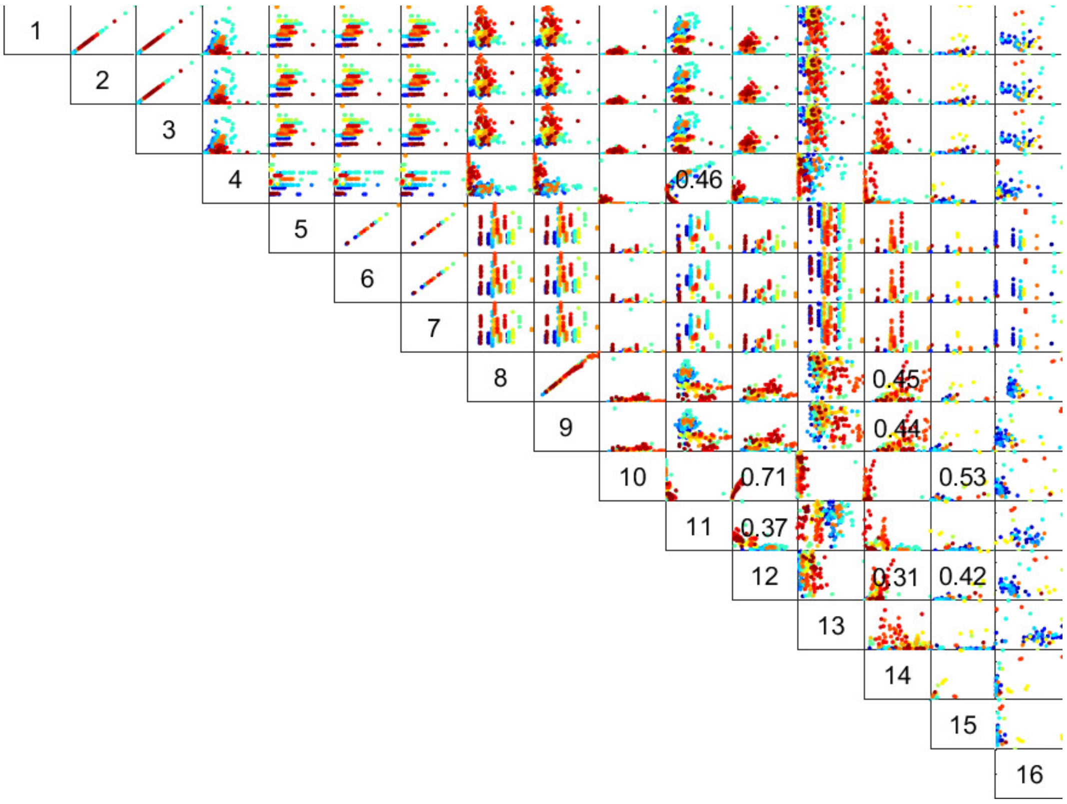


Figure B.7. Feature vs Feature for Signal 7

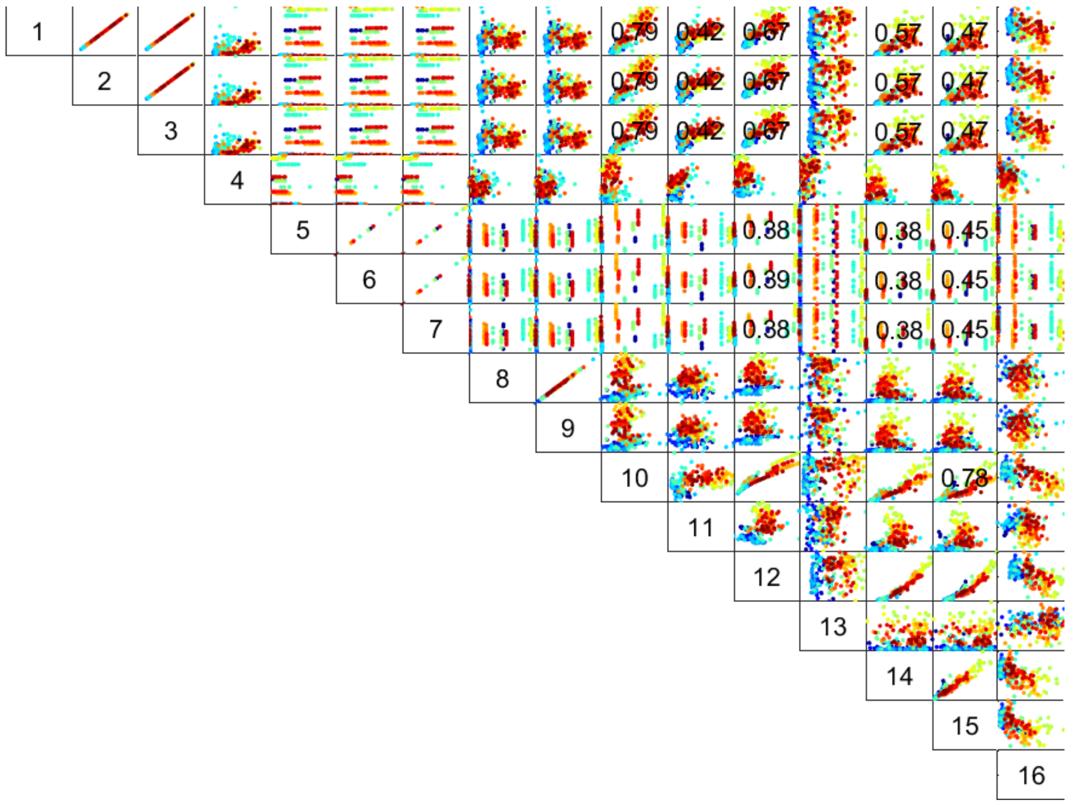


Figure B.8. Feature vs Feature for Signal 8

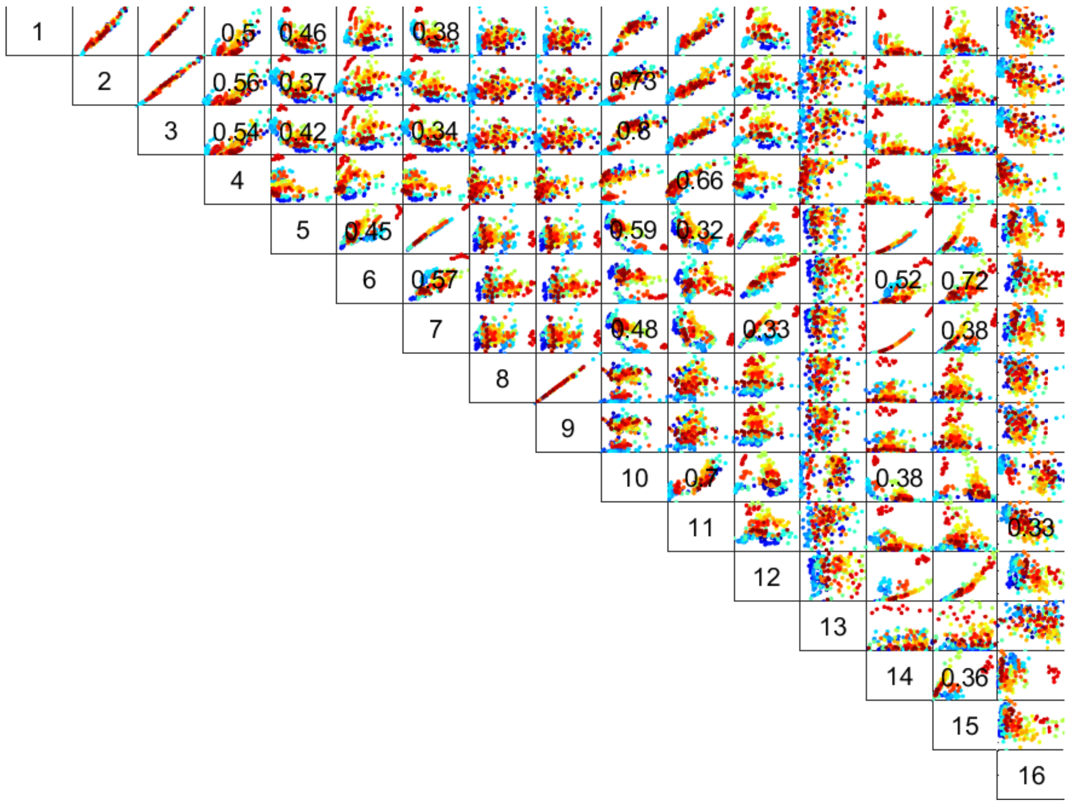


Figure B.9. Feature vs Feature for Signal 9

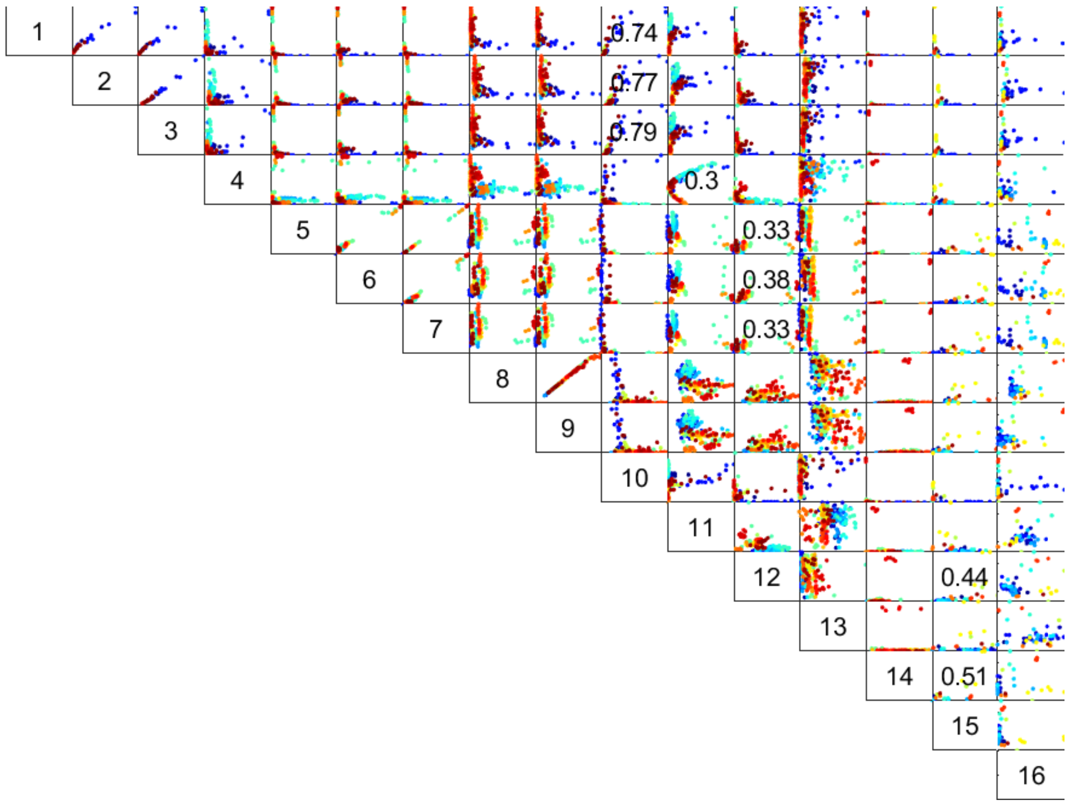


Figure B.10. Feature vs Feature for Signal 10

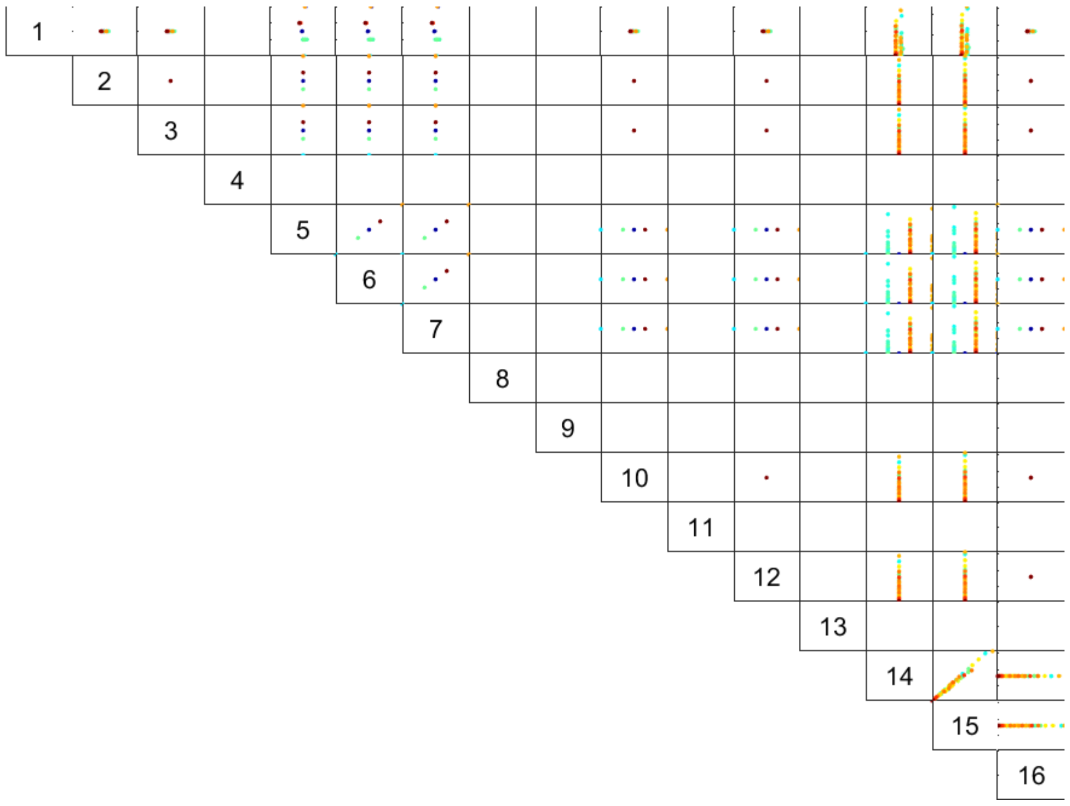


Figure B.11. Feature vs Feature for Signal 11

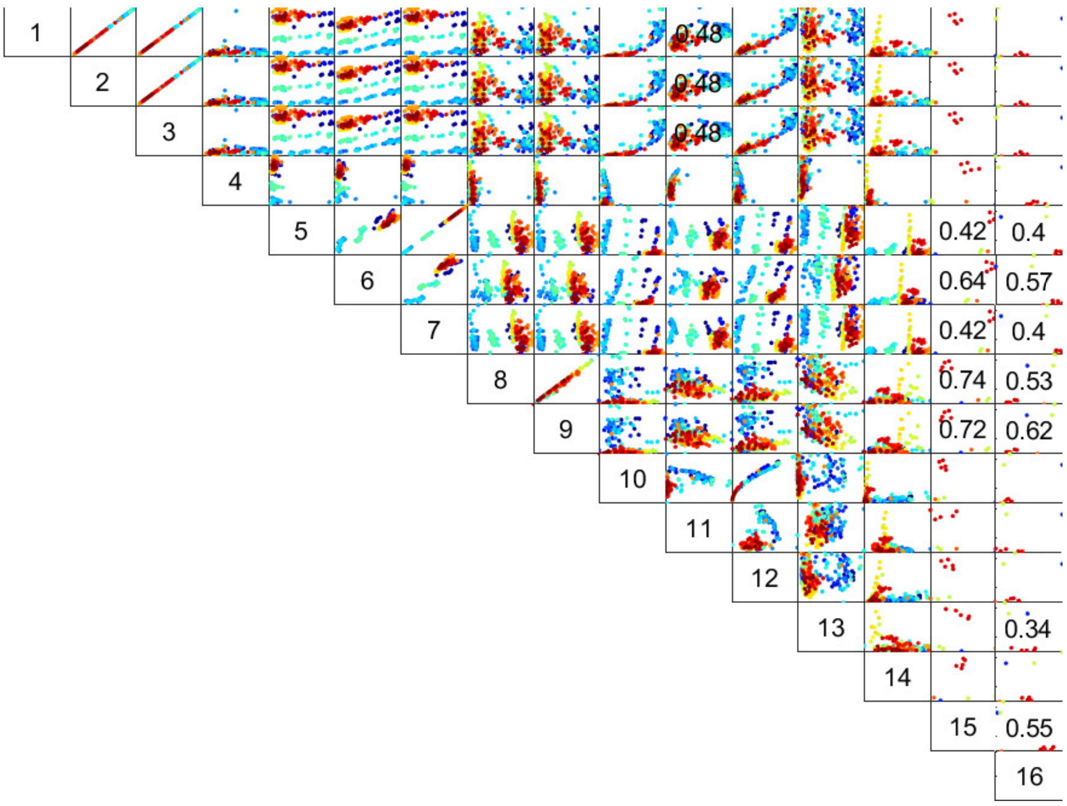


Figure B.12. Feature vs Feature for Signal 12