MÄLARDALEN UNIVERSITY
SWEDEN

School of Education, Culture and Communication
Division of Applied Mathematics

MASTER THESIS IN MATHEMATICS / APPLIED MATHEMATICS

Applying Autonomous Methods for Signal Analysis and Correction with Applications in the Ship Industry

by

Abdelghafour El Ouardi

Masterarbete i matematik / tillämpad matematik

DIVISION OF APPLIED MATHEMATICS
MÄLARDALEN UNIVERSITY
SE-721 23 VÄSTERÅS, SWEDEN

MÄLARDALEN UNIVERSITY
SWEDEN

School of Education, Culture and Communication
Division of Applied Mathematics

Master thesis in mathematics / applied mathematics

Date:
2018-06-01

Project name:
Applying Autonomous Methods for Signal Analysis and Correction with Applications in the Ship Industry

Author:
Abdelghafour El Ouardi

Supervisors:
Christopher Engström (MDH, Main)
George Fodor (QTAGG)
Farid Monsefi (QTAGG)

Reviewer:
Milica Rančić

Examiner:
Sergei Silvestrov

Comprising:
30 ECTS credits

Abstract

Tillverknings och transport industrin genererar en stor mängd datamängder, som ofta är av varierad kvalité. Målet med det här examensarbetet är att hitta matematiska principer för ett system som automatiskt lär in viktiga statistiska och analytiska egenskaper av datamängder för att detektera och korrigera vissa typer av fel i realtid.

Abstract

The manufacturing and transportation industries generate a large amount of data sets which are often of inconsistent quality. The goal of this project is to find the mathematical principles of a system which learns automatically the essential statistical and analytical properties of data sets in order to detect and correct certain classes of faults in real time.

Acknowledgements

Foremost, I would like to express my gratitude to my supervisor, Dr. Christopher Engström for the useful comments, remarks and engagement through the learning process of this master thesis. Furthermore I would like to thank Dr. Milica Rančić, Dr. Sergei Silvestrov and the participants in my survey, who have willingly shared their precious time during the process of interviewing.

I would like to express my sincere gratitude to my supervisor at QTAGG, Dr. George Fodor, for the continuous support of my MSc thesis and research, for his patience, motivation, enthusiasm, and leading me working on diverse exciting projects.
My sincere thanks also goes to Dr. Farid Monsefi, Tomas Lindqvist, Arne Löfgren, and to all QTAGG members for their hospitality.

I am so grateful to Galilee Institute at University of Paris 13 (France) and the The School of Education, Culture and Communication at Malardalen University (Sweden) for making it possible for me to study here as an exchange student. I give deep thanks to the Professors and lecturers at Galilee Institute Dr. Olivier Lafitte for his visite to QTAGG during my internship. I want to acknowledge and appreciate their help, wise advices and transparency during my research.

Last but not the least, I would like to thank my family and friends, especially Mon Trésor Asmaa, for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. I will be grateful forever for your love.

Contents

List of Figures

Chapter 1

Introduction

1.1 Mathematical Artificial Intelligence

Mathematics is at the core of AI and Machine Learning (ML) because it provides means of implementing how their goals can be reached. AI and ML automate repetitive learning and discovery through data and add intelligence to existing products. The language of mathematics is very precise. The mathematical foundation is used in physics to describe the motion of planets to the motion of electrons in atoms. It is used also in economics to describe fluctuations in the stock markets. In computer science, mathematics might be used to prove the correctness of programs. All these require a high degree of precise prediction and mathematics gives us a powerful tool, in form of optimized procedures.

The manufacturing and transportation industries generate a large amount of data which is often of inconsistent quality. For example, a large number of sensors and communication channels easily leads to situations where signals are missing, or are affected by other issues resulting in having outliers, temporary bias, etc. Detecting and correcting such problems by hand is a very tenuous task. Therefore, an unsupervised autonomous method which can learn what a good signal is and can find and correct deviations, has a large economic value.

The demand for cost-effective transportation is growing as the industrial world expands. In order to guarantee an efficient, economically safe and environmentally sustainable transportation, shipping industry has priority. Marine transportation is one of the most effective when measured in ton of transported goods per ton of fuel respectively per ton of greenhouse gases. The marine environmental impact is minimized by reducing the power effects of waves hitting the ship in order to optimize the power needed to run the ship and increasing the efficiency of the propulsion system. To predict and improve ship's stability motions and performance under various sea conditions is a critical challenge, in addition to safety of passengers.

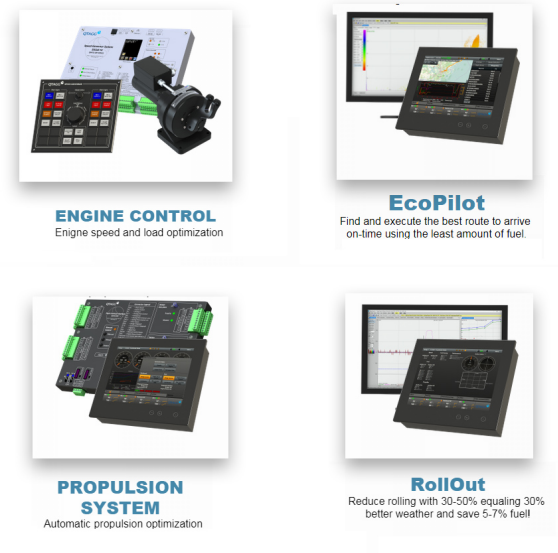


Figure 1.1: Q-TAGG's products use advanced control technologies to reduce fuel consumption on marine vessels.

1.2 QTAGG: Marine Energy Efficiency

QTAGG was founded in 1997 with the goal of bringing to the marine market a high-tech system that controls and optimizes the ship's propulsion. The company designs and installs fuel-saving solutions on large seagoing vessels. This system is governing the engine speed and fuel injection by millisecond accuracy with high precision and reduces the ship's rolling using propeller and rudder interaction. For a forthcoming voyage, it calculates an optimum propulsion plan and controls the engines and propellers to arrive on time at the lowest total cost. The system continuously analyzes the ship's motion and creates an interactive model of the physical ship and its propulsion by exposing the self-learning model to sensor data, weather forecast, and ambient parameters. The integrated Rollout system predicts the vessels motion and automatically counteracts rolling, swaying and undesirable turning before the ship is affected. QTAGG evaluates all possible scenarios by using multiple virtual ships, the alternative with lowest fuel consumption being continuously selected. In this way, the system automatically adapts to new sea conditions or other disturbances during the voyage without any manual intervention or tuning. Thus, QTAGG's control system ensures that the ship's propulsion performs at its best at all time. By reducing ship dynamics and motions, QTAGG decreases fuel consumption and decreases the wear and tear of the engine, propeller, and rudder. The diesel engine will run under a safer regime. With more information collected in time, the system will be gradually better. Figure 1.1 shows some products created by the company. Qtagg's equipment has until now (Spring 2018) six control systems:

(1) Diesel engine governor
(2) Propeller pitch control
(3) Route speed profiling and control

(4) Ship roll damping control
(5) Power generation control
(6) Propeller synchronization for vibration damping

All these systems contribute to fuel savings and have a real-time part, instrumentation, short and longtime storage, analytics, reporting and operator interface.
Experienced officers from leading shipping companies like Maersk Line, Stena Line, and Viking Line have tested and improved the system. All these systems are based on a signal collecting interface sampling continuously about 60 signals with a sample time of 250 *ms* (a data stream of about 1 kilobyte per second). Samples are often resampled to 3 minutes sample time to decrease the communications load.

1.3 Problem Description

The flow of data required in different industrial applications such as in manufacturing or in process industries has seen a sharp increase in recent years. This increase is in both in the actual data flow measured in gigabytes and in the stored historical information measured in terabytes or petabytes. Specifically for the marine industry, the increase in the data flow follows from both the increased number of sensors and from the availability of networked information from external sources such as weather reports, marine traffic reports and from information generated by different systems onboard and on-shore. It is usual that sensor information is distributed among all systems onboard using the NMEA 0183 (National Marine Electronics Association) communication standard.

Anybody who experienced a large factory with thousands of sensors knows that signals are not always perfect. There are interruptions in the data flow, disturbances, outliers, interferences, line brakes, etc. Traditionally, such situations are handled by alarm systems which are supervising that signals are continuous and in their normal range. When an alarm occurs, a human operator is reading the error messages from a central alarm system and is trying to detect and correct the error. Manual error detection and correction requires that the operator knows a great deal about the systems generating the signal and about a set of known hypothesis for why an error has occurred. This method has however limited use on a ship. The crew on a ship is usually reduced to the number of people who can navigate the ship and the expertize in signal processing, system analysis, network debugging, etc., might be in short supply onboard a ship.

In other industrial applications such as manufacturing, the situation is not as limited as in marine applications, but still the size of the problem is an issue to be handled : the number of signals on an industrial plant might be in the range of millions and finding and correcting errors can be a challenge. Another aspect is that signals that are used in real-time control applications are required to be of high quality. Real-time control systems are acting at millisecond time granularity and a wrong signal can cause a faulty control effect which can result in damages to the equipment before any human intervention could be done.

Autonomous signal monitoring systems are therefore becoming subjects of major interest for the industry in general and specifically for the shipping industry. They enable automatic detection of abnormalities and can correct abnormalities by having information from other related signals. Finally, monitoring systems can help in order to identify the faulty equipment to be replaced. Mathematically, the autonomous operations can be explained as follows. Let us consider that a ship (an industrial plant) has N signals obtained from sensors at each sample time instance t: $s_1, s_2, ..., s_N$. These signals have normally a degree of redundancy such that for example the signal s_p can be estimated from say three other signals s_i, s_j and s_k such that:

$$\hat{s}_p(t) = f_p(s_i(t), s_j(t), s_k(t)) \tag{1.1}$$

where the 'hat' over the variable s_p means "estimation" and the function f_p is considered known. By knowing the estimation function, one can compare the difference ε:

$$\varepsilon = max||s_p(t) - \hat{s}_p(t)|| \tag{1.2}$$

When the absolute value of ε becomes large, it means the signal $s_p(t)$ is different from the estimated value and this might mean a signal fault. In such a situation, the values of $s_p(t)$ can be replaced by $\hat{s}_p(t)$. This method works if the estimation function and its parameters are known to be correct. In principle, the method to automate detecting and correcting signals works as follows:

- Learn the essential properties of the signals
- Use these properties to detect if signals are correct
- Find all possible correlations and relationships between signals
- Check continuously that the estimated value of a signal from the correlation function does not differ much from the measured value
- If the estimated value differs from the measured value and all the signals and functions in the estimation are correct, then replace the value of the signal with the value of the estimation

The research problem defined as described above, does not have a well-defined field. There are many different results in signal processing, ML, function identification, sensor fusion, fault detection, identification, etc. which can be used to solve this problem. Therefore, we call the problem defined in this thesis as *Signal Curation*. This also poses difficulties in finding reliant approach literatures. Some of the approaches considered in this thesis are described below.

The methods applied during the thesis are most often of statistical nature. One distinction is made between statistical methods and ML methods. In the statistical modeling approach, it is assumed that data is generated by a process with identifiable statistical properties such as mean or standard deviation. In ML approaches, there are no assumptions about the generation process (which is considered a black-box), but the data is modeled as such. ML approach is used in studying a vast sample of measurements to detect characteristics patterns or relationships which were not previously taken into account. It is true that expert knowledge performs a good diagnostic (tedious when dealing with dozens of signals), but ML is seen as a complementary and effective approach to identify the group of signals the most likely to influence

another one.

We use data from experiments of an ocean-going vessel to study the abnormal behavior of signals. The experiments have been done on a container ship between 2017-12-01 and 2018-01-22. An automated data collection device (Q-TAGG Energy Evaluation Device) sampled data on the ship. The resulting logs files are sampled at 3 minutes interval obtained from raw data, a fragment is shown in Figure 1.2. The table has about 60 columns and about one million samples. The logged data is organized in columns such that signals have to originate from physical sensors used by the energy optimization equipment, data from the ship's NMEA data network and indirect measurements computed from propeller etc.

Figure 1.2: Raw Data from Ship..

Chapter 2

Literature Review

It is a difficult task to find the relevant literature in autonomous mathematical methods for signal analysis and correction since in general mathematics is involved in many research fields related to signal analysis and thus there are numerous papers related to this subject. For the purpose of delimiting the literature, we have classified the existing research into the following subfields:

1. Traditional signal processing methods
2. Fault detection and isolation (FDI) based on mathematical approaches
3. Sensor fusion methods
4. Health monitoring systems

Traditional signal processing methods are using continuous and discrete methods to improve the quality of a signal for its intended purposes. Typical applications are signal filtering, signal conditioning, outlier detection and removal and signal analysis. The later include different decomposition methods such as Fourier, Laplace or wavelet transforms.
Signal conditioning means preparing a signal such that it is adapted to the next stage in a hierarchical processing stage, for example a digital-to-analog conversion or an impedance conversion.
Signal filtering means removing from the signal components such as high or low frequency noises which are not describing the intended process generating the signal. A reference book in this subject is for example [1] and another good reading is the Matlab reference signal processing manual [2].
The difference between a traditional research paper in signal processing and our approach is the following. Traditional signal processing deals with the properties of one signal. The purpose of the signal processing function is a well defined function from start, such as filtering or digital conversion, where most often the performance of the filter in terms of execution time, induced delays or costs is the most important factor. By contrast, in our approach the precise processing of the signal is not known a-priori and an intelligent mathematical algorithm must find an appropriate processing using both other signals and historical data about the signal.
The computational costs are not an important factor but the autonomous property of the processing is important. In short, we seek an automated processing where an algorithm can

determine and apply in an intelligent way a processing.

While in signal processing the device generating data is not considered, in FDI methods the generating unit is essential. In FDI it is considered that one can extract a model of the process generating the data and can find and correct faults in this generating device. This is a very important function in applications.
There is a noticeable difference between research papers in FDI and Signal Processing: in general, FDI methods are targeting a certain type of industrial process while the Signal Processing methods are general. The International Federation of Automatic Control (IFAC) has a working group in FDI technology, with so called "benchmark" applications in actuator FDI, chemical FDI, stirred tank heater FDI, wind turbines FDI, etc [3]. A reference book in FDI is [4]. Typical mathematical methods used in FDI are state estimation, model identification, symptom representation, signature methods and parity equations.
The difference between the FDI approach and our approach is that we seek mathematical methods to compensate (correct) signals but we are not seeking to find the faults in the process generating the signal. This means our approach has a smaller and different scope than FDI.

The sensor fusion approach has origins in military applications from the US Joint Directors of Laboratories. The method combines several signals to improve information about a target signals in the same way as our approach does, however the focus is on the computer architecture and the processing steps in each architecture. The architectural details of the sensor fusion approach are not so relevant for our work, however there are similarities between the methods applied in Sensor Fusion, our approach and in general in AI and ML. For example the paper [5] uses covariance matrices in the same fashion as presented in this work. The article [6] uses Bayes, Markov and random set methods which are close to our approach .

The approach pursued in this thesis is close to the one described in the article "Data Mining of Flight Measurements", where the authors explain the importance of health monitoring systems for the jet engine industry as they enable to detect abnormalities in the behavior for an engine in order to schedule a maintenance action in advance. Data Mining consists in studying a vast sample of measurements to detect characteristic patterns or connections which were not previously taken into account. This article presents a processing sequence which, without any a priori knowledge is able to detect phenomena that had not been observed in the training set. This processing sequence is designed to treat vast amount of data and offers the possibility of a physical interpretation - or verification - of built models. In order to predict a given variable, the most consistent choice always consists in selecting the variables sharing the more information with the variable we are interested in. The definition of independent subsets can then be performed by unsupervised classification methods [7].

Chapter 3

Method of Analysis

3.1 Overall approach

Mathematically, each signal is a function of one variable describing how some physical quantity varies over time. According to reference [2], the fuel consumption of a ship, for example, depends on some technical factors such as engine efficiency, hull and propeller state. It depends also on the environmental conditions such as wind, waves and sea current. A ship sailing on the same route could have fuel consumption variations in the range of 30-100 % due to changes in sea and weather conditions. The ship acceleration and heading control have an immediate impact on the fuel consumption. Thus, the fuel consumption depends on the following signals:

- Ship speed in $[kn]$
- Rudder Pos = rudder angle $[deg]$
- RollAmp = roll amplitude $[deg]$
- Roll Pitch = ship's roll pitch $[deg]$ (more pitch means more waves and more resistance)
- WindX = wind component along the ship in $[kn]$
- WindY = wind component across the ship in $[kn]$
- DepthFact = 1/(Depth+20) sea depth factor, it is the inverse of depth
- Slip = propeller slip in $[\%]$.Computed with the formula:

$$s = 1 - \frac{30.86666 \times V}{P \times N} \tag{3.1}$$

where P = propeller pitch $[m]$ and N = engine rotation speed $[rot/min]$.
(the factor 30.86666 is from conversion of $[kn]$ to $[m]$ and [rot/sec] to [rot/min], that is $0.514444 \times 60 = 30.866664$).

When it comes to regression methods, that are subsequently used, it is not a good approach to use too many predictors. Adding irrelevant signals can potentially disrupt learning process and bury helpful variables influence as shown in the Figure 3.1.

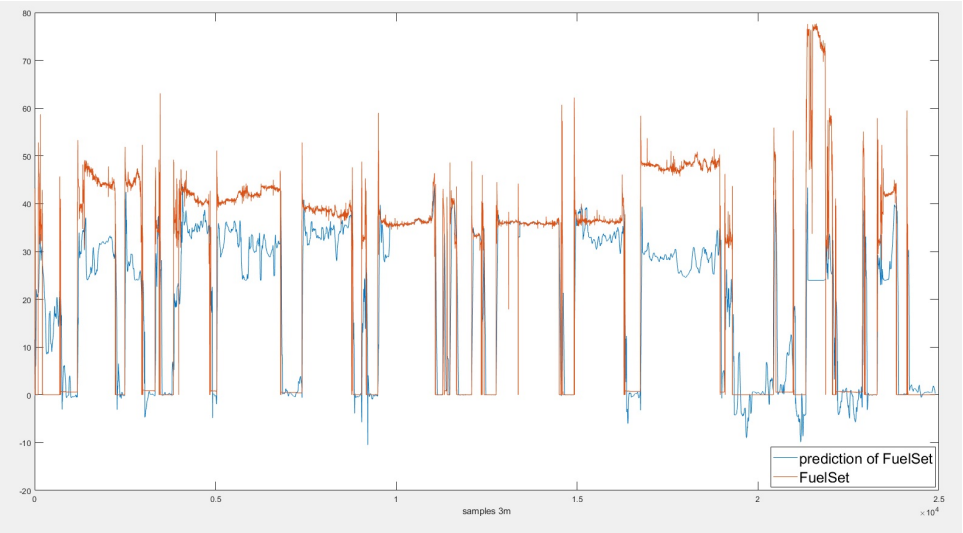


Figure 3.1: Predictioin of Fuel consumption using V, RudderPos, RollPitch, Wind and Depth-Fact as predictors.
Using Regression Learner in MATLAB, the training process continues until the model achieves a desired level of RMSE (the square root of the variance of the residuals) on the training data. The Gaussian Process Regression (GPR) algorithm is used. Too many predictors applies a huge error betwen the Fuel consumption signal and its prediction.

Before talking about the identification of coherent groups of signals, we introduce some operations to correct individual signals without relation to other signals. First, we classify the signals according to their sign (positive signals, negative signals or positive and negative signals) and their nature (real or relative integer). By doing this, we could specify if the signal does not change over time, so it is a stationary function. We can consider that the signal is stationary if the amplitude is almost constant, it means that the value of the signal is always within a fixed band as in Figure 3.2. In this case, if the signal becomes a steep function (constant over time), that means we have certain deviation and we must correct the error. We also detect an error if we find real numbers on a signal whose values must be integers.

To improve one single signal, we may identify its disturbances using filtering and outlier detection and removal. For example, a ship has the Speed Over Ground (SOG) and Speed Through Water (STW) as shown in Figure 3.3. It can be seen that the SOG has a sensor fault: some samples have spikes down to zero and some other samples have smaller spikes upwards. Clearly, the ship cannot have such sudden variations in speed. A so-called Hampel algorithm identifies and corrects these type of outliers. The Hampel filter block detects and removes the outliers of the input signal by using the Hampel identifier. For each sample of the input signal, the block computes the median of a window composed of the current sample and $\frac{Len-1}{2}$ adjacent samples on each side of the current sample. *Len* is the window length we specify through the Window length parameter. The block also estimates the standard deviation of each sample about its window median by using the median absolute deviation. If a sample differs from the median by more than the threshold multiplied by the standard deviation, the

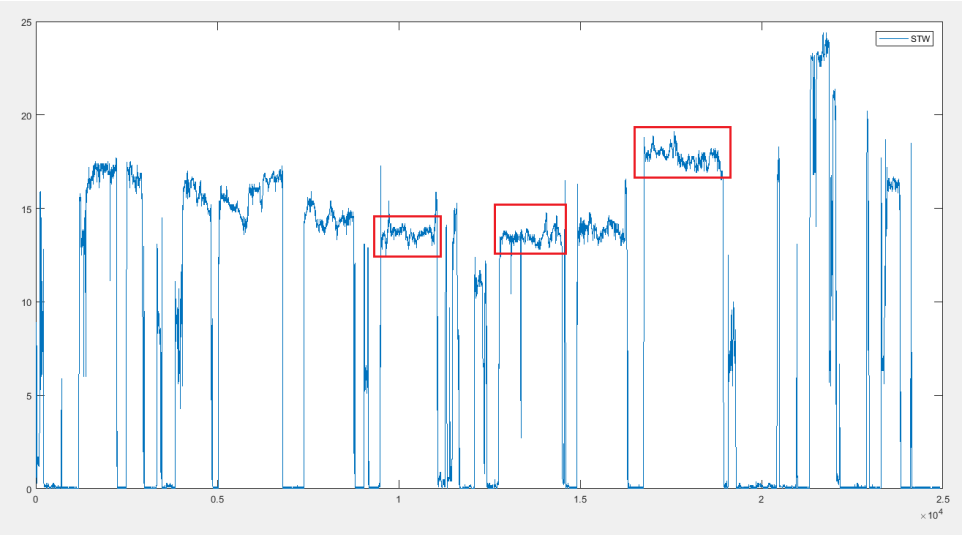


Figure 3.2: Example where the signal is considered stationary.

filter replaces the sample with the median [9].
A data point may not be an outlier even if it is located far away from other measurements. However, if the signal contains significant outliers, we may need to consider the use of robust statistical techniques. In general, for a given sample data x_s, the Hampel algorithm:

- Centers the window of odd length at x_s.
- Computes the local median m_i and standard deviation σ_i over the current window.
- Compares the current sample with $\alpha \times \sigma_i$ where α is the threshold value.
- Identifies the current sample x_s as an outlier if $|x_s - m_i| > \alpha \times \sigma_i$.
- Replaces x_s with the median value m_i.

The corrected filtered values are in red color in Figure 3.3. It can be seen that the green and red values are precisely identical at samples where there are no outliers, but the spikes are removed and replaced with interpolated values. The Hampel algorithm is tuned by knowing the maximum acceleration a ship can achieve in normal conditions (estimated from displacement of the ship, friction and water and air surface).

Qtagg has about 20 different filters and transformations that are applied to correct signals. Some are simple, like traditional filters, but some others require advanced mathematics. The correction is done either using information about individual signals or using redundancy to correct one signal using other signals which are more correct. Figure 3.4 shows another example with faults found in STW measurements [8]. Traditional filtering of a signal is used for rolling, pitching, generated current etc. An example of rolling angle filtering is shown in Figure 3.5 [8].

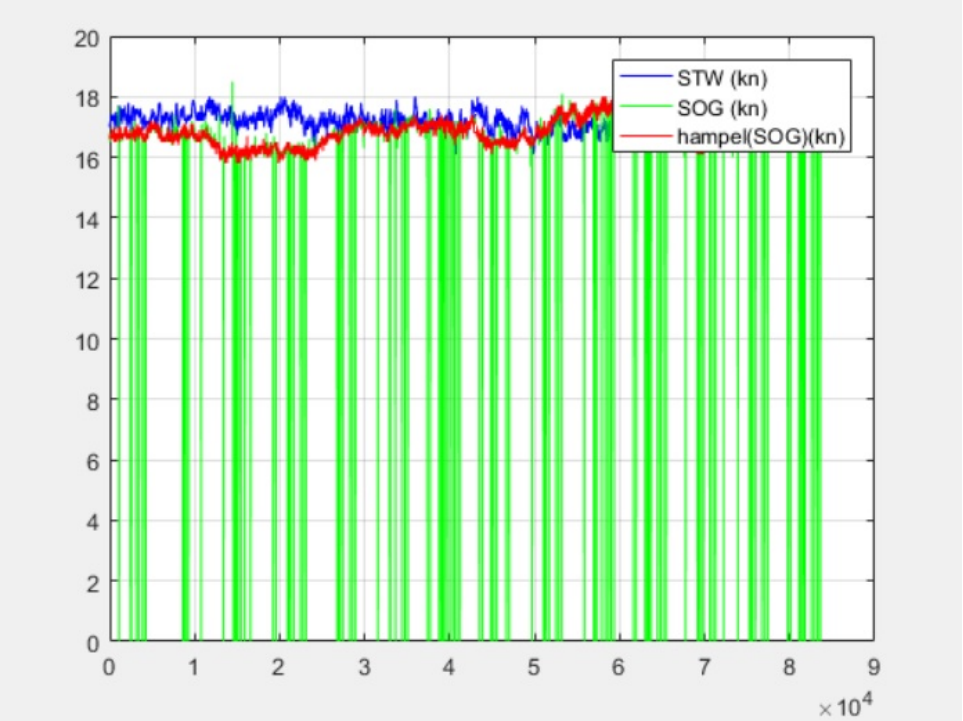


Figure 3.3: Outlier detection and compensation example.

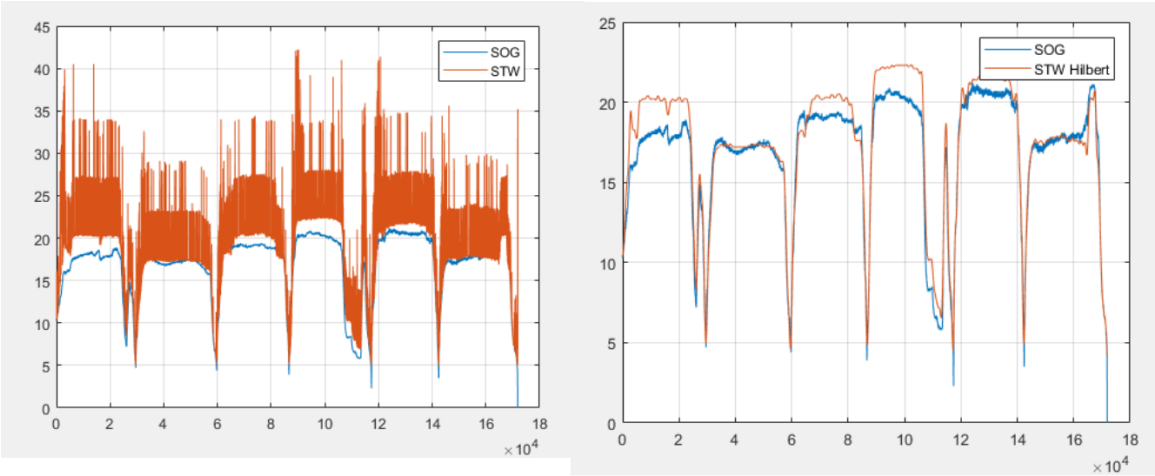


Figure 3.4: Noisy STW with spikes - before and after conditioning.
The STW signal (in orange) is very noisy and has many spikes upwards from the actual value. A so called Hilbert envelope transform recovers the correct signal.

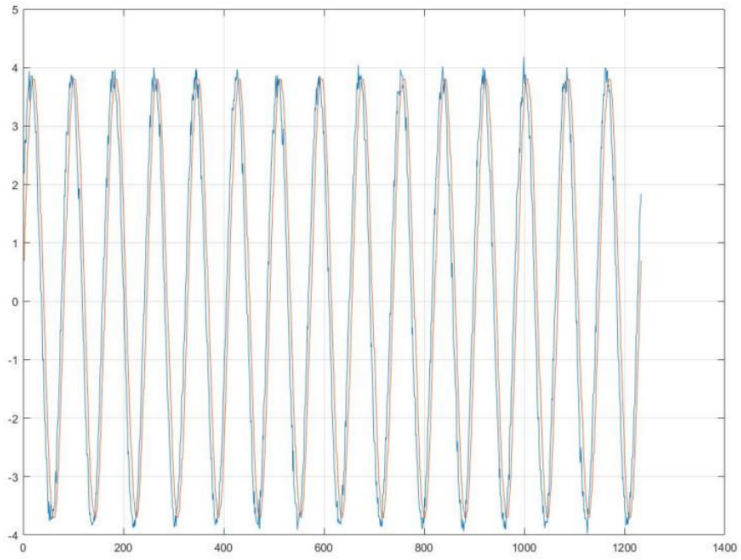


Figure 3.5: Roll angle filter.
It can be seen that the signal has a phase shift that needs to be considered in the model-predictive controllers.

3.2 Kalman Filter

Another way to correct a signal is to use Kalman filter which is an optimal solution to many data prediction tasks. Also known as linear quadratic estimation (LQE), Kalman filter is an algorithm that uses a series of measurements observed over time, containing statistical noise and other inaccuracies, and produces estimates of unknown variables that tend to be more accurate than those based on a single measurement alone, by estimating a joint probability distribution over the variables for each timeframe. The filter is named after Rudolf E. Kalman, one of the primary developers of its theory [10].
In order to extract the required information from a signal, we calculate the difference between this information x_k and its estimation $\hat{x}_k$ which is termed the error

$$f(e_k) = f(x_k - \hat{x}_k) \tag{3.2}$$

A signal can be described by

$$y_k = a_k x_k + n_k \tag{3.3}$$

where y_k is the observation, a_k is a gain term, x_k is the information bearing signal and n_k is the additive noise. The function $f(e_k)$ should be positive and it increases monotonically. An error function which respects these characteristics is the squared error function

$$f(e_k) = (x_k - \hat{x}_k)^2 \tag{3.4}$$

The expected value of the error function, so-called the loss function ε is given by

$$\varepsilon(t) = E(f(e_k)) = E(e_k^2) \tag{3.5}$$

since it is necessary to consider the ability of the filter to predict many data over a period of time. Using maximum likelihood statistics, we redefine the goal of the filter to finding the $\hat{x}_k$ which maximizes the probability of y:

$$max(P(y|\hat{x})) \tag{3.6}$$

with

$$P(y|\hat{x}) = \prod_k P(y_k|\hat{x}_k) \tag{3.7}$$

We assume that the additive random noise is Gaussian distrubuted with a standard deviation of σ_k:

$$P(y_k|\hat{x}_k) = K_k exp(-\frac{(y_k - a_k\hat{x}_k)^2}{2\sigma_k^2}) \tag{3.8}$$

where K_k is a normalization constant. The maximum likelihood function is given by:

$$P(y|\hat{x}) = \prod K_k exp(-\frac{(y_k - a_k\hat{x}_k)^2}{2\sigma_k^2}) \tag{3.9}$$

which leads to:

$$logP(y|\hat{x}) = -\frac{1}{2}\sum_k (\frac{(y_k - a_k\hat{x}_k)^2}{\sigma_k^2}) + constant \tag{3.10}$$

The driving function of this equation is the mean squared error MSE, which may be maximised by the variation of $\hat{x}$.

Here we present the Kalman Filter Recursive Algorithm [11]. We consider that we want to know the value of a variable within a process of the form:

$$x_{k+1} = \phi x_k + w_k \tag{3.11}$$

where x_k is the state vector of the process at time k, ϕ is the state transition matrix of the process from the state at k to the state at $k+1$, and w_k is the associated white noise process with known covariance. We can model this observation by

$$z_k = Hx_k + v_k \tag{3.12}$$

where z_k is the actual measurement of x at time k, H is the noiseless connection between the state vector and the measurement vector, v_k is the associated measurement error which is assumed to be a white noise process with known covariance and has zero cross-correlation with the process noise. We assume that the covariances of the two noise models w_k and v_k are stationary over time and are given by:

$$Q = E[w_k w_k^T] \; ; R = E[v_k v_k^T] \tag{3.13}$$

The error covariance matrix P_k at time k is given by :

$$P_k = E[e_k e_k^T] = E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T] \tag{3.14}$$

which is equivalent to the definition of the mean squared error MSE.
We consider $\hat{x}'_k$ the prior estimate of $\hat{x}_k$ and we define an update equation for the new estimate, combing the old estimate with measurement data:

$$\hat{x}_k = \hat{x}'_k + K_k(z_k - H\hat{x}'_k) \tag{3.15}$$

where K_k is the Kalman gain which will be derived shortly and $i_k = z_k - H\hat{x}'_k$ is the measurement residual. Using (2.12) and (2.15) we have

$$\hat{x}_k = \hat{x}'_k + K_k(Hx_k + v_k - H\hat{x}'_k) \tag{3.16}$$

We replace in the MSE:

$$P_k = E[e_k e_k^T] = E[(x_k - (\hat{x}'_k + K_k(Hx_k + v_k - H\hat{x}'_k)))(x_k - (\hat{x}'_k + K_k(Hx_k + v_k - H\hat{x}'_k)))^T] \tag{3.17}$$

$$P_k = E[[(I - K_kH)(x_k - \hat{x}'_k) - K_kv_k][(I - K_kH)(x_k - \hat{x}'_k) - K_kv_k]^T] \tag{3.18}$$

with $x_k - \hat{x}'_k$ is the error of the prior estimate, which is uncorrelated with the measurement noise:

$$P_k = (I - K_kH)E[(x_k - \hat{x}'_k)(x_k - \hat{x}'_k)^T](I - K_kH)^T + K_kE[v_kv_k]K_k^T \tag{3.19}$$

$$P_k = (I - K_kH)P'_k(I - K_kH)^T + K_kRk_k^T \tag{3.20}$$

using $R = E[v_kv_k^T]$ with P'_k is the prior estimate of P_k we end up with the error covariance update equation. The diagonal of the covariance matrix contains the mean squared errors:

$$P_{kk} = \begin{bmatrix} E[e_{k-1}e_{k-1}^T] & E[e_ke_{k-1}^T] & E[e_{k+1}e_{k-1}^T] \\ E[e_{k-1}e_k^T] & E[e_ke_k^T] & E[e_{k+1}e_k^T] \\ E[e_{k-1}e_{k+1}^T] & E[e_ke_{k+1}^T] & E[e_{k+1}e_{k+1}^T] \end{bmatrix}$$

The mean squared error may be minimized by minimizing the trace of P_k and so minimizing the trace of P_{kk}. We have

$$P_k = (I - K_kH)P'_k(I - K_kH)^T + K_kRk_k^T \tag{3.21}$$

$$P_k = P'_k - K_kHP'_k - P'_kH^TK_k^T + K_k(HP'_kH^T + R)K_k^T \tag{3.22}$$

$$trace(P_k) = trace(P'_k) - 2trace(K_kHP'_k) + trace(K_k(HP'_kH^T + R)K_k^T) \tag{3.23}$$

Differentiating with respect to K_k gives:

$$\frac{\partial trace[P_k]}{\partial K_k} = -2(HP'_k)^T + 2K_k(HP'_kH^T + R) \tag{3.24}$$

The result set to zero in order to find the conditions of the minimum which represents the Kalman gain equation:

$$K_k = P'_kH^T(HP'_kH^T + R)^{-1} \tag{3.25}$$

Finally, the error covariance equation is given by :

$$P_k = P'_k - K_kHP'_k - P'_kH^TK_k^T + K_k(HP'_kH^T + R)K_k^T \tag{3.26}$$

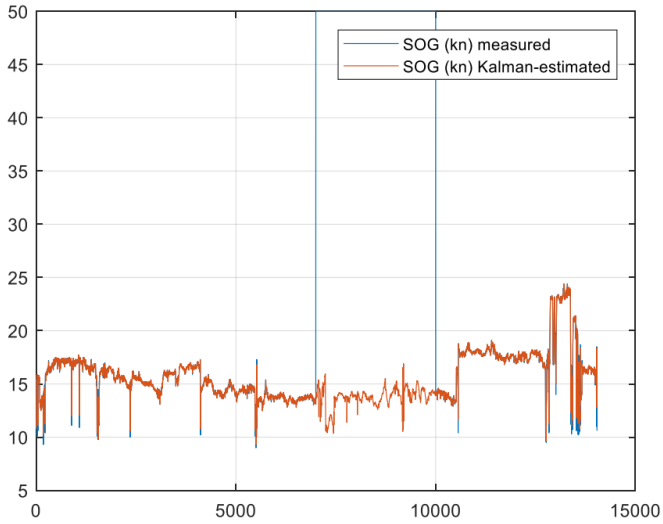


Figure 3.6: Kalman filter estimation of SOG.
A Kalman filter identifies the fault and corrects the SOG (shown in orange color).

$$P_k = P'_k - K_k H P'_k \tag{3.27}$$

$$P_k = (I - K_k H) P'_k \tag{3.28}$$

The last equation is the update equation for the error covariance matrix with optimal gain. State projection is achieved using

$$\hat{x}'_{k+1} = \phi \hat{x}_k \tag{3.29}$$

We complete the recursion by projecting the error covariance matrix into the next time interval $k+1$. First, we form an expression for the prior error:

$$e'_{k+1} = x_{k+1} - \hat{x}'_{k+1} \tag{3.30}$$

$$e'_{k+1} = (\phi x_k + w_k) - \phi \hat{x}_k \tag{3.31}$$

$$e'_{k+1} = \phi e_k + w_k \tag{3.32}$$

According to equation (2.14) the error covariance matrix at time $k+1$ is given by:

$$P'_{k+1} = E[e'_{k+1} e^T_{k+1}] = E[(\phi e_k + w_k)(\phi e_k + w_k)^T] \tag{3.33}$$

Since the noise w_k accumulates between k and $k+1$, e_k and w_k have zero cross-correlation.

$$P'_{k+1} = E[\phi e_k (\phi e_k)^T] + E[w_k w_k^T] = \phi P_k \phi^T + Q, \tag{3.34}$$

And this completes the recursive filter. Figure 3.6 shows the case when a signal freezes due to a communication error: the blue signal SOG is lost due to a short-term error in the GPS, the value is frozen at 50 kn, which is a speed clearly over the normal range for a ship. However, the STW and environment signals are available from parallel measurements. Kalman filters, when the signal interruption is not too long, can restore perfectly all missing samples.

Chapter 4

Signal Processing

Signal analysis for classification is used in different industries like electronics, aerospace, defense, finance, automotive etc. In our problem, we have N signals and we want to predict the type of each signal. What we mean by type here is the physical quantity represented by the signal: STW or Wind or Fuel Set Average etc. To do so, we use signal processing to minimize the impact of noise and interference on the signal.

In this section, we will see how signal processing methods can be applied to preprocessing signals and to extracting descriptive features. To design a method that can learn from pieces of information of logs, we are going to use some statistical measurements. Here we present a brief study of two signals. We plot first STW and Wind signals to visualize how they're distributed in time.

In Figure 4.1, we see that the wind signal is oscillating much then STW signal. We could quantify this by looking at the statistics of the signal. Using histogram, we could easily separate Wind and STW signal as shown in Figure 4.2 based on standard deviations or root mean squared value. But what about if we had to work out the difference between STW and SOG signals? Figure 4.3 shows that both signals have a similar average value and similar standard deviation. In this case, we should consider some more advanced analysis of how values vary over time by looking at the rate of the oscillations for example.

Signal Processing Toolbox allows us to analyze, visualize and compare multiple signals and detect and extract features or interesting events. Now the question is : can we measure how quickly our signals are oscillating? many signal processing tools are used to quantify the shape or fingerprints of these oscillations. A good choice would be by looking at the spectral presentaion of the signals. To estimate the power spectral density (PSD), we use Welsh method which is popular. In general, the goal of spectral estimation is to describe how the power contained in a signal is distributed over frequency. Mathematically, PSD of a stationary random process $x(n)$ is related to the autocorrelation sequence R_{xx} (correlation of a signal with a delayed copy of itself as a function of delay) by the discrete-time Fourier transform:

$$P_{xx}(\omega) = \frac{1}{2\pi} \sum_{m=-\infty}^{\infty} R_{xx}(m) e^{-j\omega m}. \tag{4.1}$$

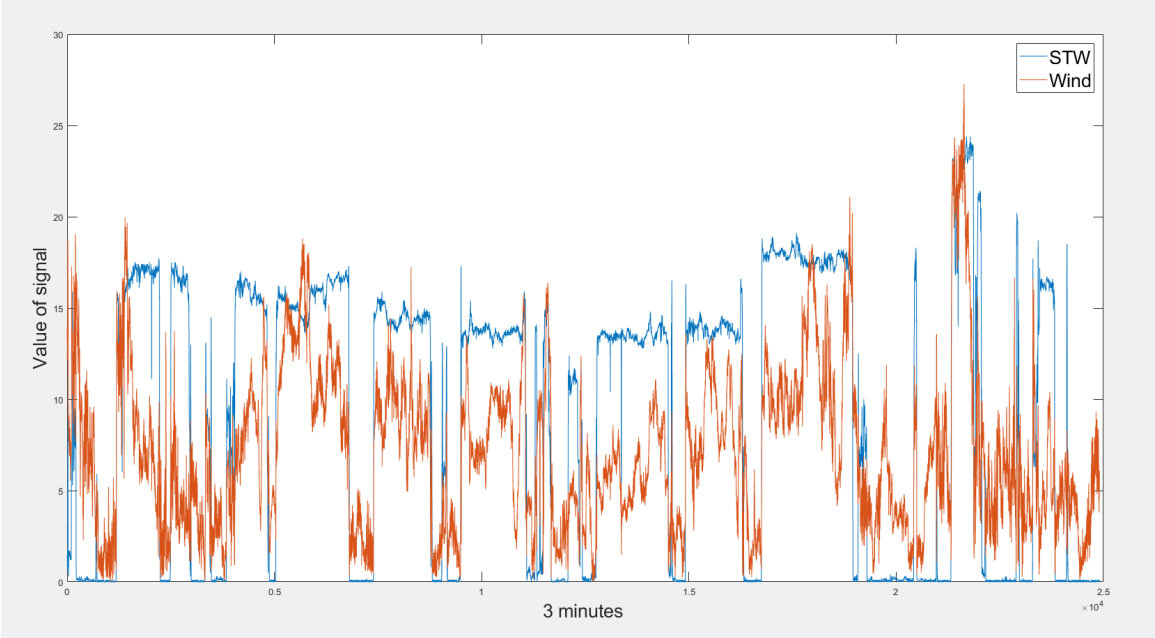


Figure 4.1: Speed Through Water STW and Wind.

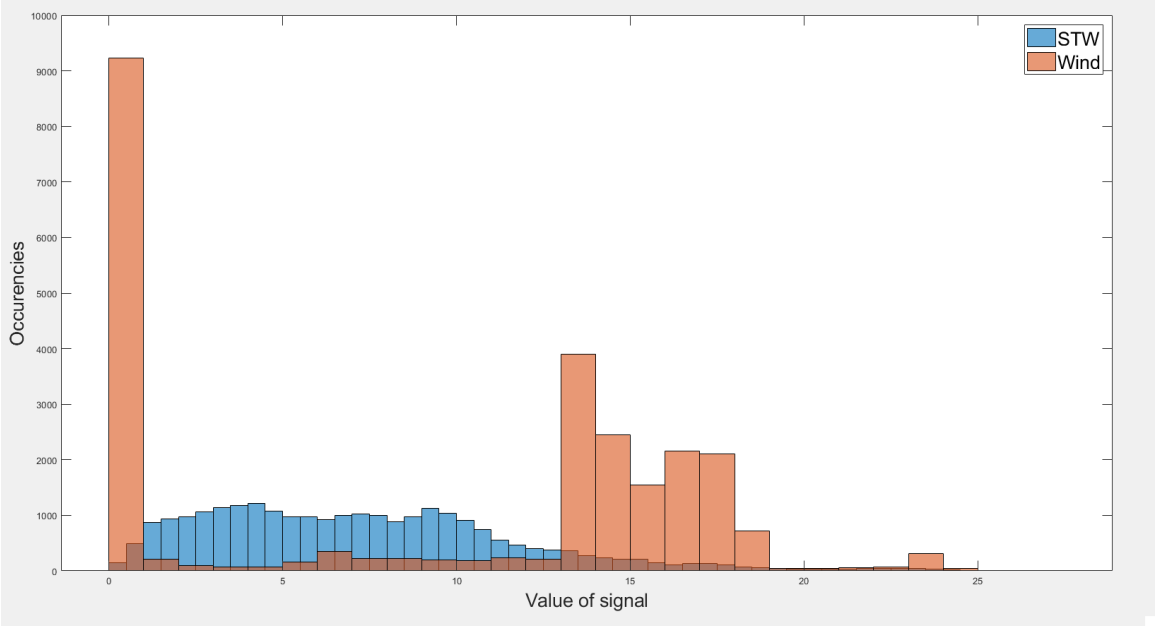


Figure 4.2: Speed Through Water STW and Wind histogram.

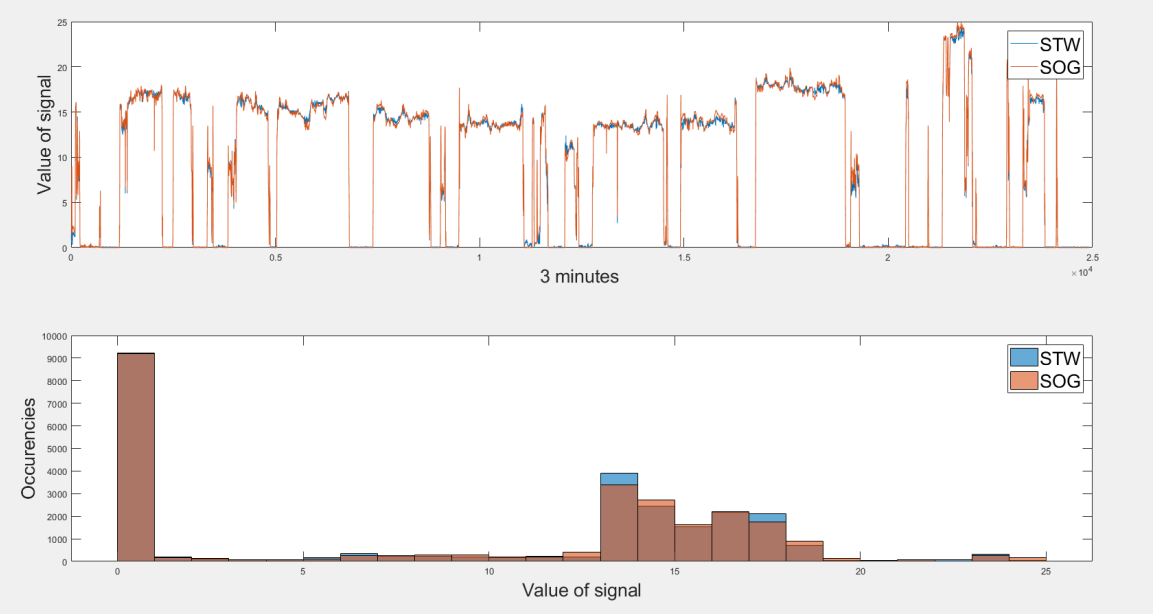


Figure 4.3: Curve and histogram of STW and SOG signals.
Discrimination of STW and SOG using statistical analysis is not sufficient

We can see in Figure 4.4 how the frequency varies as a function of time. The power of the signal is available in form of colors such that the blue color means low power levels and yellow color depicts high power levels.

We apply the filter [12] to our signals that create new signals where we hope to find only the contributions due to physical quantity measured by the signal. In Figure 4.5, we plot both STW and Wind signals. The new signals now are centered around zero. Now we look at the spectral representation using the Welch method. In Figure 4.6, we have the frequency on the x-axis from 0 to half of the sampling frequency which was 50 Hertz, and the y-axis we have the power density in units Db over Hertz. The region when the values of this plot are higher is likely to carry the information that we are after. In this case, the pattern of peaks between 0 and 1.5 Hertz is holding a lot of measurable information on the rate shape of the time domain oscillations. The distance in frequency between peaks tells us about the rate of oscillation of signals and the relative amplitudes are closely related to the shape of the oscillations. Using the MATLAB function *findpeaks*, we specify locations and amplitudes of some peaks. The positions and the amplitudes of peaks carry descriptive quantitative information which if measured would constitute good descriptive features.

Extract Features is used to minimize the loss of important information embedded in signals, minimize the complexity of implementation and reduce the cost of information processing. Signals on a ship are expected to be robust, have high-quality and low disturbances. However, signals in practice have often limitations. Fuel flow meters are not precise due to transport delays and fuel return effects, STW sensors have sensor-related disturbances, wind measurements on the ship are biased by the height of the sensor, there are communication failures,

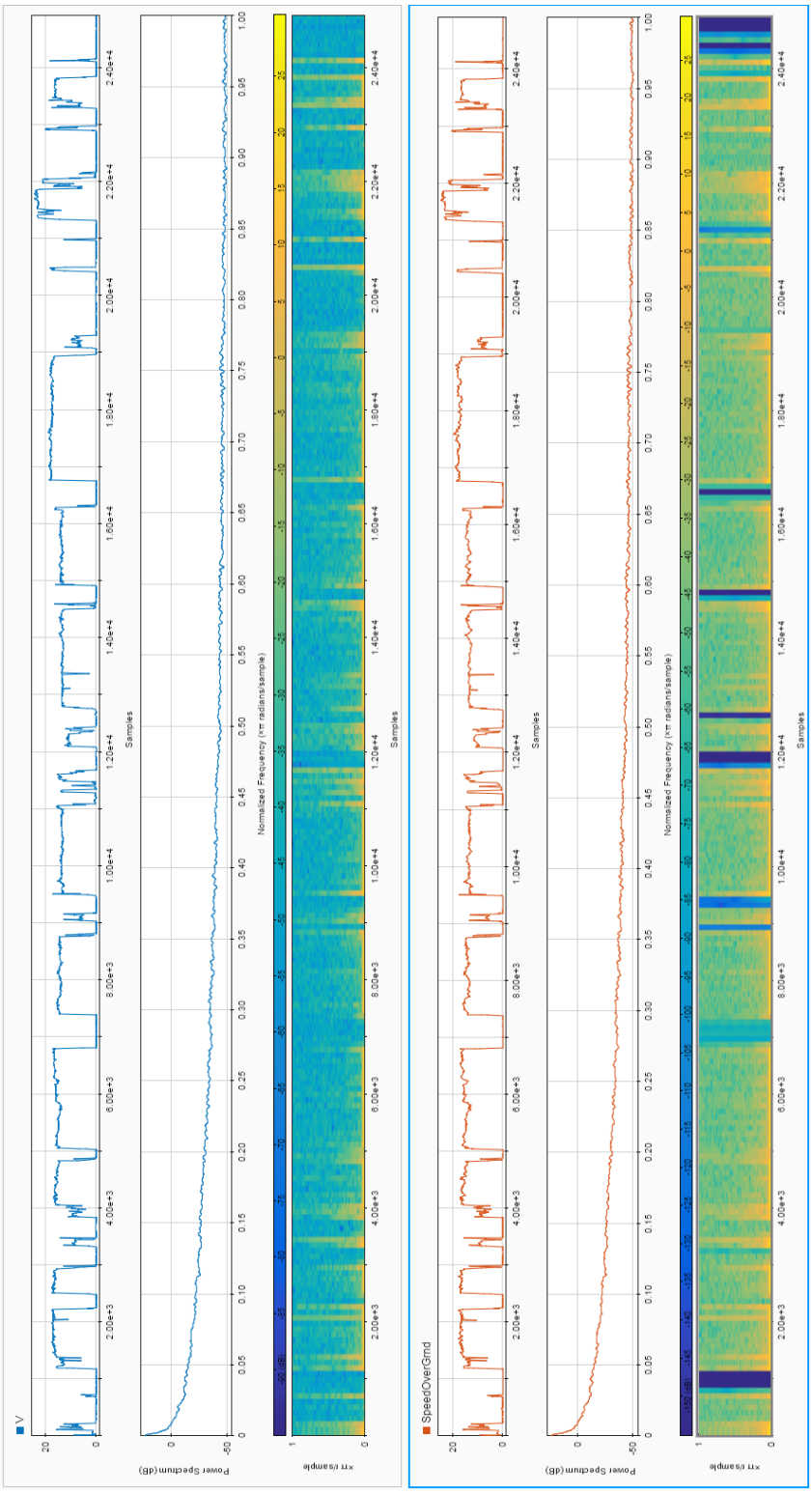


Figure 4.4: Spectrogram of STW and SOG.
Visual representation of the spectrum of frequencies of the two signals STW and SOG as they vary with time. Here we can clearly see the difference between the two signals in the frequency domain

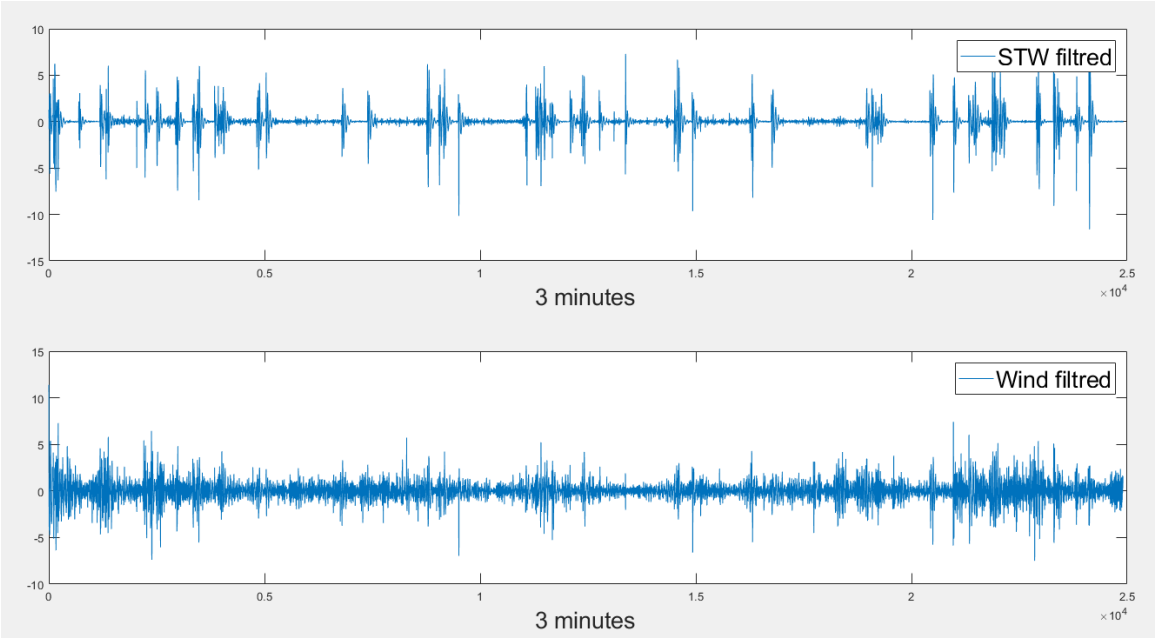


Figure 4.5: STW and Wind filtered signals. The two signals are centered around zero

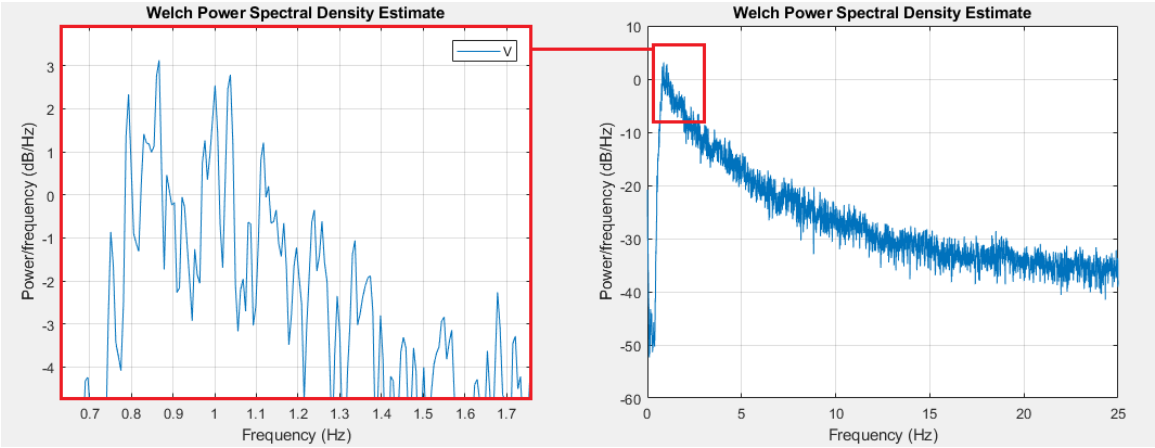


Figure 4.6: Welch power spectral density estimate.

unexplained outliers, and noise on all signals, etc. This is precisely where ML methods start to be part of the picture.

Chapter 5

Machine Learning

5.1 Introduction

ML is the use of algorithms to create knowledge from data. As shown in Figure 5.1, problems such as speech recognition, object recognition, and engine health monitoring are too complex for handwritten rules or equations. We can use ML algorithms to learn such complex nonlinear relationships. Weather forecasting, energy load forecasting, and stock market prediction are examples of applications where the underlying system is constantly changing, so the solution also needs to change and adapt. We can quickly learn from new data using ML and thus keep up with dynamic systems. Applications like Internet of Things (IoT) analytics, taxi availability, and analysis airline flight delays involve typically learning from large amounts of data and ML algorithms are designed to learn efficiently from such large sets [12].

5.2 Overview and workflow

In simple terms, ML uses data and produces a program to perform a task. Let us take an example of human activity detection using a smartphone [12]. We use the internal tri-axial accelerometer to determine when the person is performing activities like sitting, standing and walking. In the traditional approach, we might be able to use some handwritten rules based on empirical observations or to try to develop a formula. The solution will be suboptimal in either case because of the complexity of the problem since the solution space has many combinatorial possibilities. A ML solution would involve the collection of large amounts of sensor data with corresponding activity levels, then allowing the algorithm to learn these complex nonlinear relationships between inputs and outputs. The challenge here is to select the most appropriate algorithm and avoid pitfalls such as overfitting.

According to Figure 5.2, we can say that ML can be divided into supervised and unsupervised learning. In supervised learning, the data sets includes input values as corresponding outputs. Our task is to train a model to predict or estimate the output on a new data. If the output has discrete value, we say that we have a classification problem. If the output is continuous, we refer it as a regression problem. For example, if we are trying to predict a signal

Figure 5.1: World Examples of Machine Learning and AI.
These images are captured from Maths Work separately and we combined them into a single image

type (Wind or STW etc) then it is a classification problem. If we are trying to predict a ship's displacement (tones) from its dimensions, then it is a regression problem. Both examples are discussed in this thesis.
In case of unsupervised learning, our task is to group the data based on some measure of similarity, and input data set does not have corresponding output or labels and we want to discover naturally occurring patterns in the data. Clustering techniques fall under the category of unsupervised learning. An example of unsupervised learning is the problem of determining correlated or more general related signals so that missing information is continuously detected and corrected.

Figure 5.3 presents the steps involved in a typical ML workflow. We begin by accessing and exploring the data. In this step, we prepare data from different sources like various file formats, databases or even streaming data from sensors. The pre-processing is the second step in which raw data is manipulated by cleaning signals of messy data example outliers or missing values, extracting features or predictors and applying various data reduction and transformation techniques to keep only the useful information. Once data is preprocessed, we develop predictive models, train and compare multiple models, optimize parameters and validate models performance to ensure robustness. The final step is to share and deploy train models by integrating them into analytic pipelines. Models may run on embedded systems shared as standalone applications or even deploy on the cloud.

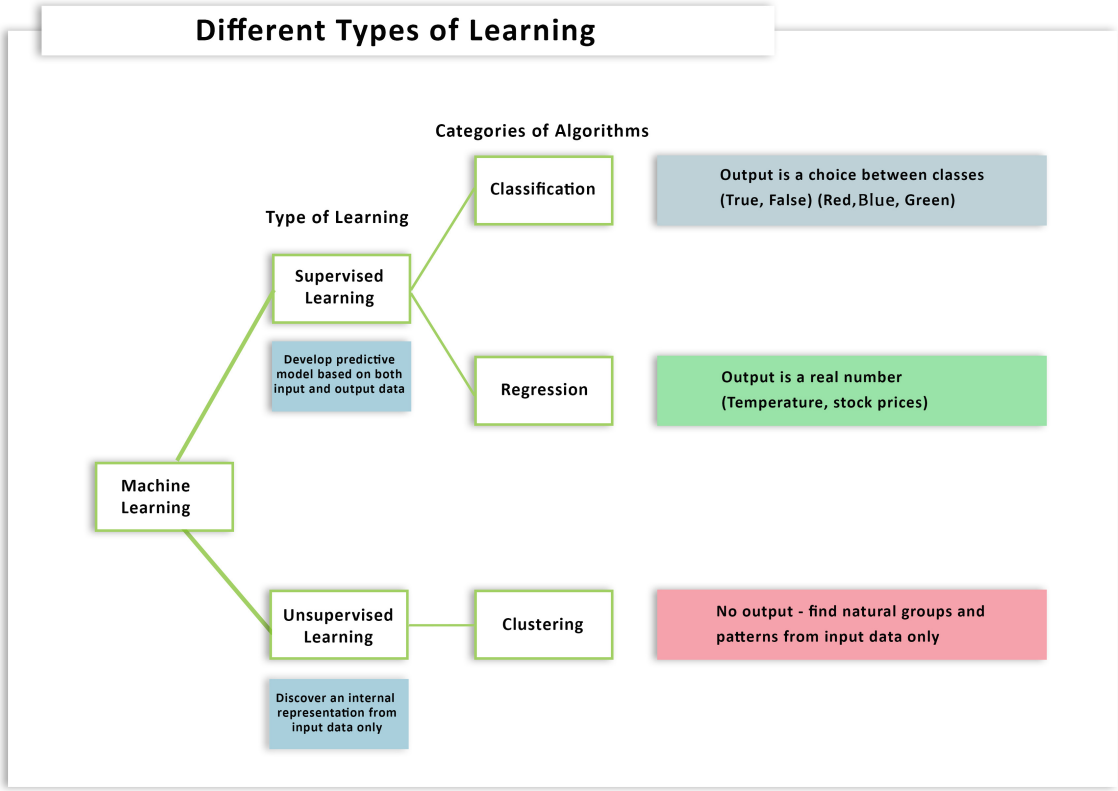


Figure 5.2: Some types of Machine Learning Algorithms.

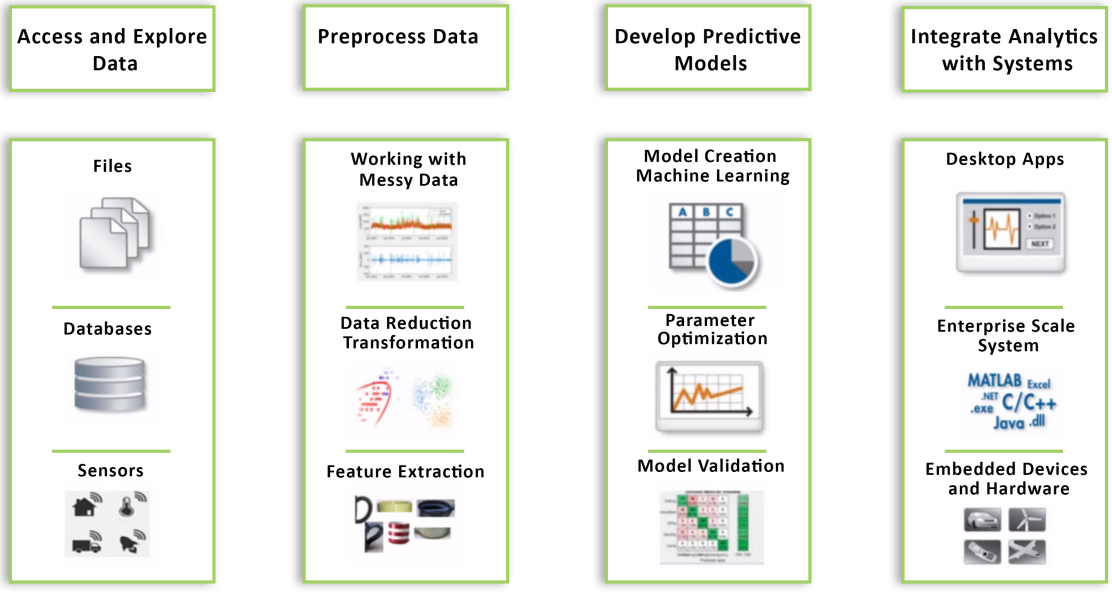


Figure 5.3: Workflow steps.
These images are captured from Maths Work separately and we combined them into a single image

5.3 Classifications of Signals

ML is the science of getting computers to act without being explicitly programmed. The goal of ML is to develop methods that can detect some information automatically in data, then to use the uncovered information to predict future data. One example of the supervised learning is the classification method, which is used to learn a mapping from inputs x to outputs y, where y is the number of classes. To make predictions on novel inputs, we can formalize the classification method as function approximation. We assume $y = f(x)$ for some unknown function f, and we want to estimate the function f by given a labeled training set, and then we make a prediction using $\hat{y} = \hat{f}(x)$. As a simple example of classification, consider that we have two uncorrelated signals: STW and Roll Pitch, which correspond to label 0 and 1. The inputs are the signal characteristics. These have been described by a set D of features: mean, variance, maximum and minimum value of signals, which are stored in a $N \times D$ design matrix X shown in the following table:

D features	max	min	mean	var	label
STW	24.4000	0	9.1435	56.9606	0
Roll Pitch	1.3000	0	0.1956	0.0457	1

We assign a class (or label) from a finite set of classes to an observation. Our goal is to identify if a new signal is a STW or Roll Pitch signal. We need to predict a continuous measurement for an observation to answer this question. For this reason, instead of those two observations, we assume that we have 100 logs of data. One way to do so is to add 100 normally distributed random numbers to each value in the previous table so we end up with 100 different observations.

D features	max	min	mean	var	label
STW	24.4000	0	9.1435	56.9606	0
STW	24.1660	0	10.7268	58.2546	0
...	...	...	...	...	...
STW	23.3533	0	8.8941	57.3226	0
Roll Pitch	1.3000	0	0.1956	0.0457	1
Roll Pitch	1.0369	0	0.0842	1.8270	1
...	...	...	...	...	...
Roll Pitch	2.3821	0	1.4916	0.3975	1

Using Classification Learner in MATLAB, the training process continues until the model achieves the desired level of accuracy on the training data. The Simple Tree algorithm is used to go from observations about an item to conclusions about item's target. It is one of the predictive modeling approaches used in statistics, data mining and ML.
Because of the important difference between characteristics values of the two signals, our program can identify easily which is the STW and Roll Pitch signal of a new unknown signal. Now we check the algorithm with four different signals and see if we can classify a new signal. In this case, we suppose that we have the following signals: STW, Roll Pitch, Wind, and Fuel Set Average. The signals characteristics are given by the following table:

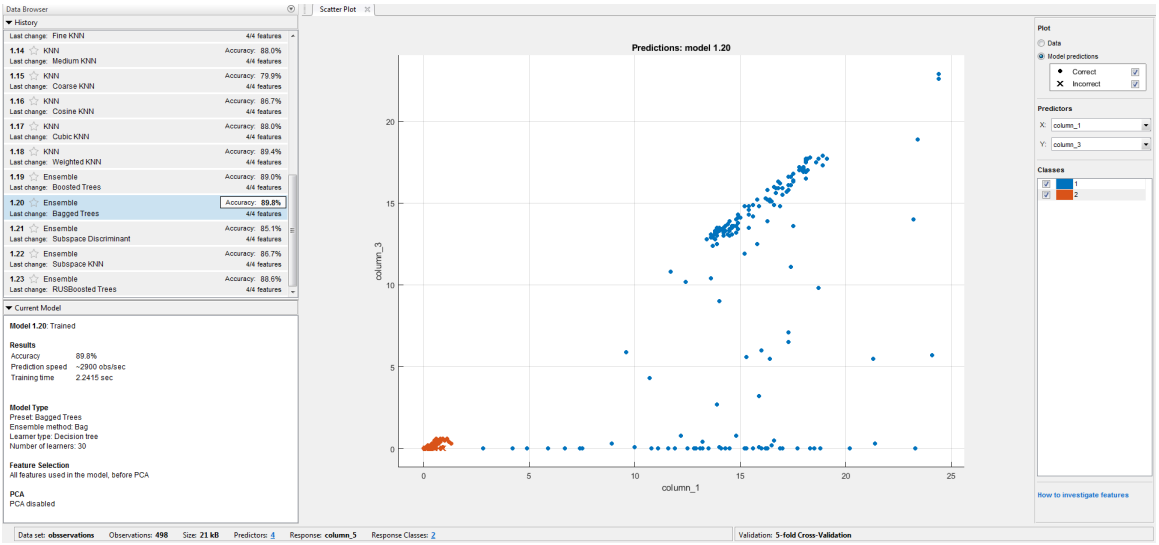


Figure 5.4: Ensemble Bagged Trees Algorithm, the case of two signals

D features	max	min	mean	var	label
STW	24.4000	0	9.1435	56.9606	1
Roll Pitch	1.3000	0	0.1956	0.0457	2
Wind	27.2776	0	7.2052	19.2204	3
FuelSetAve	77.7000	0	25.5884	437.3954	4

It can happen that our algorithm cannot distinguish the two signals STW and Wind since the values of the maximum and mean of those two signals are close to each other. Now we test the algorithm by using new logs, i.e, a new data set of information in our program, we can classify all signals except Wind, it returns the label corresponding to STW. This requires the need to add more features using signal processing to classify correctly our signals. An initial conclusion is that simple statistical analysis is often not sufficient as we see in the previous section. We can use an other approach to get many observations from data by splitting data into intervales of 100 samples and extracting features from each subinterval. In this case, we use Ensemble Bagged Trees Algorithm since the model achieves the desired level of accuracy on the training data as shown in the Figure 5.5. We can predict the type of a new signal in this case also. Now we use again a set of four signals: STW, Roll Pitch, Wind, and Fuel Set Average. We use Ensemble Bagged Trees algorithm again. To predict the type of a new signal, we split this signal into intervals of 100 samples and try to predict the type of each subinterval and calculate how long we have a good answer, then we choose the signal with a higher number of correct predictions as a final prediction answer. In this case, Figure 5.5, our algorithm can predict the label of all signals.

We have other classification techniques:

- Logistic Regression
- K Nearest Neighbors

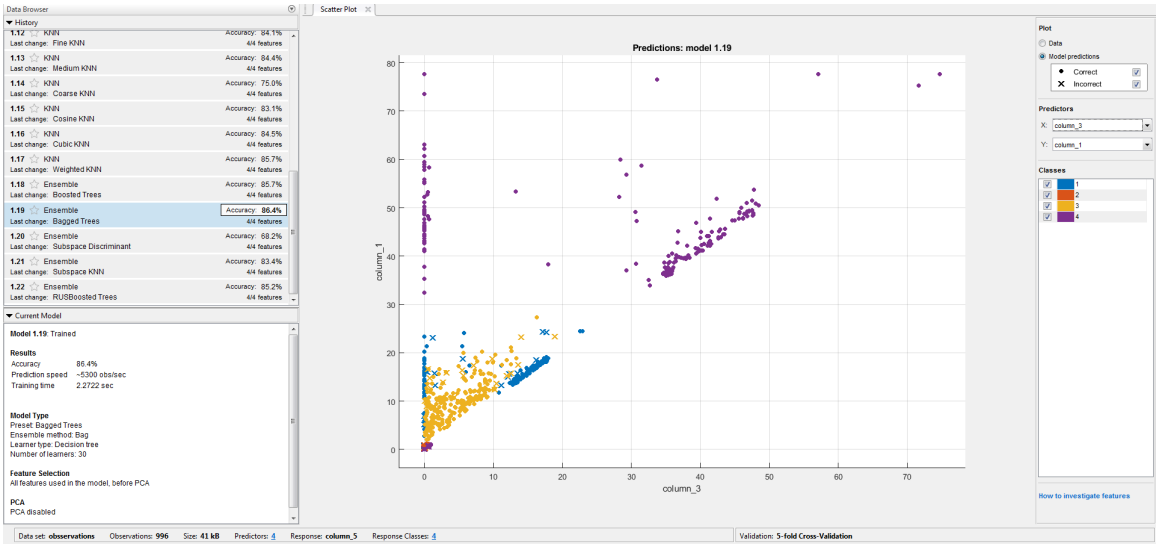


Figure 5.5: Ensemble Bagged Trees Algorithm, the case of four signals

- Decision Trees
- Naïve Bayes
- Neural Networks
- Support Vector Machines

The first step is that the algorithm is trained with a large set of known or labeled data optimizing its free parameters to identify those known cases as accurately as possible. Once trains, it can be run on unknown new signal to formulate a prediction on what's the most likely the class for that new signal. We can use Support Vector Machines (SVM) as a model that best splits the data into two different sections. As showns in Figure 5.6, the hyperplane best splits the data because it is as far as possible from support vectors. So, we have maximized the margin by solving a constrained optimization problem. We first define the decision rule. Consider ω, a vector constrained to be perpendicular to the hyperplane and u a vector that points to some unknown data point. We are interested in whether or not our unknown data point is on the right side or on the left side of the hyperplane. To do so, we measure $\omega.u$ and see whether or not that number is equal to or greater than some constant c : $\omega.u \geq c$
The decision rule is given by: if $\omega.u+b \geq 0$ with $b=-c$, then the unknown data point is on the left side of the hyperplane (without loss of generality).
We don't know what constant b to use and we don't know which ω to use either. We have $\omega.u_1+b \geq 1$ if we take the unknown data point as sample from the first section and $\omega.u_2+b \leq -1$ if the vector u is pointing to a sample belongs the second section. We introduce a variable y_i, such that $y_i = 1$ in the first group and $y_i = -1$ in the second group. Then we multiply both equations by y_i

$$y_i(\omega.u_1+b) \geq y_i \; ; \;\; y_i(\omega.u_2+b) \leq -y_i \tag{5.1}$$

Which is equivalent to

$$y_i(\omega.u_1+b) \geq 1 \; ; \;\; y_i(\omega.u_2+b) \leq -1$$

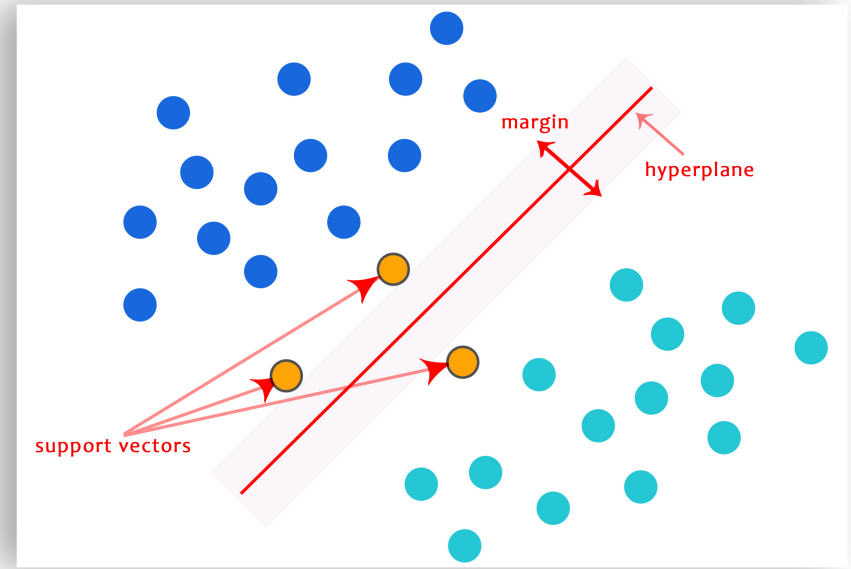


Figure 5.6: The hyperplane separates two classes with the maximum margin.

$$y_i(\omega.u+b) \geq 1$$
$$y_i(\omega.u+b)-1 \geq 0$$

and $y_i(\omega.u+b)-1=0$ if u is a support vector.
The margin is given by $(u_1-u_2)\frac{\omega}{||\omega||}$.
Here we assume that u_1 and u_2 are two vectors pointing to supports vectors (orange data points in the schema). If we have a data point from the first section, so $(\omega.u_1+b)-1=0$ since $y_i=1$, and then $\omega.u_1=1-b$. We have $\omega.u_2=-1-b$ if the data point is from the second section. Finally, the margin is given by $(1-b-(-1-b))\frac{1}{||\omega||}=\frac{2}{||\omega||}$. We want to maximize $\frac{2}{||\omega||}$ which is equivalent to minimize $||\omega||$ or minimize the quadratic optimization problem $\frac{1}{2}||\omega||^2$.
We use Lagrange multipliers to find the extremum of a function with constraints, which gives us a new expression that can be minimized without thinking about constraints any more.

$$L=\frac{1}{2}||\omega||^2-\sum\alpha_i[y_i(\omega.u_i+b)-1]$$
$$\frac{\partial L}{\partial \omega}=\omega-\sum\alpha_i y_i u_i=0 \implies \omega=\sum\alpha_i y_i u_i$$

The decision vector ω is a linear sum of support vectors.

$$\frac{\partial L}{\partial b}=-\sum\alpha_i y_i=0 \implies \sum\alpha_i y_i=0$$

We replace the value of ω in L

$$L=\frac{1}{2}||\sum\alpha_i y_i u_i||^2-\sum\alpha_i[y_i((\sum\alpha_j y_j u_j).u_i+b)-1]$$

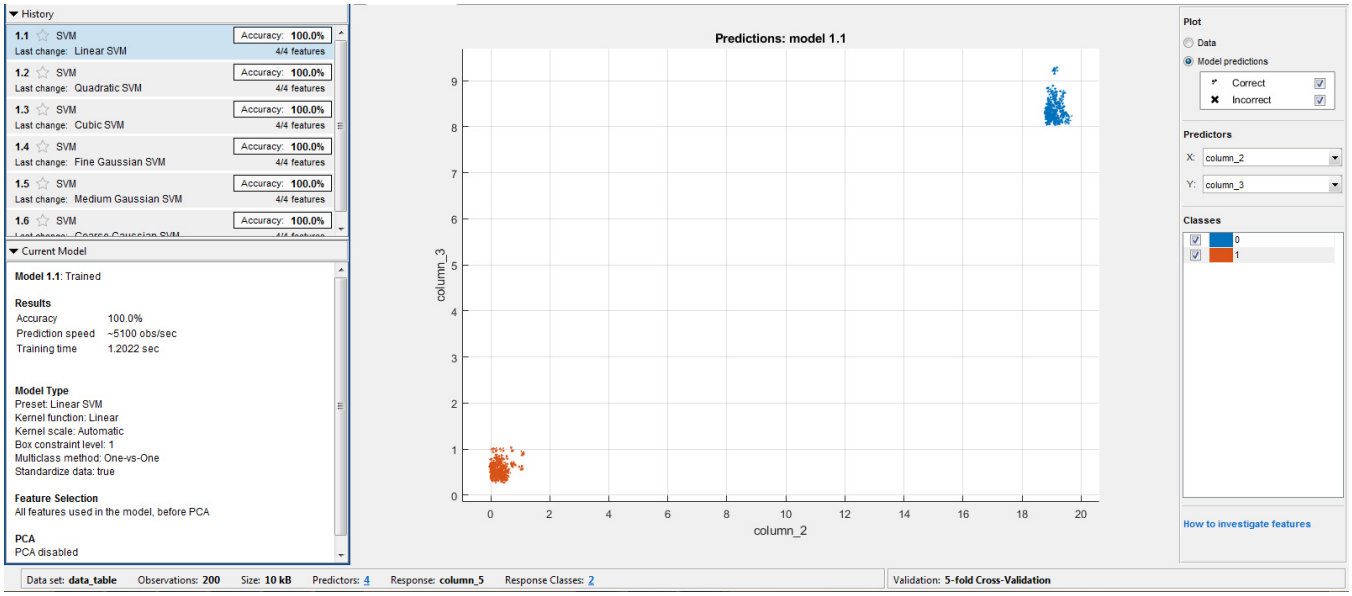


Figure 5.7: Scater Plot: variance vs maximum.

$$\implies L = \frac{1}{2}(\sum \alpha_i y_i u_i)\ (\sum \alpha_i y_i u_i) - \sum \alpha_i y_i u_i (\sum \alpha_j y_j u_j) - \sum \alpha_i y_i b + \sum \alpha_i$$

$$\implies L = \frac{1}{2}(\sum \alpha_i y_i u_i)\ (\sum \alpha_j y_j u_j) - \sum \alpha_i y_i u_i (\sum \alpha_j y_j u_j) - b\sum \alpha_i y_i + \sum \alpha_i$$

$$\implies L = \frac{1}{2}(\sum \alpha_i y_i u_i)\ (\sum \alpha_j y_j u_j) - \sum \alpha_i y_i u_i (\sum \alpha_j y_j u_j) + \sum \alpha_i\ ; (\sum \alpha_i y_i = 0)$$

$$\implies L = -\frac{1}{2}\sum \alpha_i y_i u_i\ \sum \alpha_j y_j u_j + \sum \alpha_i$$

$$\implies L = \sum \alpha_i - \frac{1}{2}\sum\sum \alpha_i \alpha_j y_i y_j\ u_i.u_j$$

So, the optimization depends only to the dot product of pairs of samples $u_i.u_j$.
The decision rule is given finally by: If

$$\sum \alpha_i y_i u_i.u + b \geq 0$$

then the unknown data point is on the left side of the hyperplane. Thus, The decision rule depends only on the dot product of those sample $u_i.u$ Consider agin that we have two uncorrelated signals: STW and Roll Pitch signals. When given a new signal, our model should automatically predict whether that signal is a STW or Roll Pitch signal. The inputs are the signal characteristics. These have been described this time by a set of features: mean, variance, maximum and standard deviation. Before setting our model, we visualize the data as shown in Figure 5.7 (variance 'column 2' vs max 'column 3') Using SVM algorithm, and because of the important difference between characteristics values of the two signals, the program can identify easily which is the STW and Roll Pitch signal of an unknown signal.
We are able to separate the two signals features using a line since it is a 2-dimensional plot.

In 3 dimensions we could separate data points with a plane. In higher dimension (Hard to visualize) we would separate data with hyperplane. The reason that we want to add in more dimensions is because with more dimensions there is the opportunity to further maximize the margin and create a better model. SVM model able to classify more than two signals. We need to chose carefully the useful features used to represent a signal.

5.4 Regression methods

5.4.1 Gaussian Process Regression

Gaussian Process Regression (GPR) works by extending the idea of a probability distribution of numbers to a probability distribution of functions. The one-dimensional form of the Gaussian distribution is given by

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{5.2}$$

where μ is the mean, σ is the standard deviation, and σ^2 is the variance.
The multivariate version of the previous equation is given by

$$N(x\,;\mu,\Sigma) = \frac{1}{(2\pi)^{N/2}|\Sigma|^{1/2}} exp[-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)] \tag{5.3}$$

where $x, \mu \in \mathbf{R}^N$. $\Sigma \in \mathbf{R}^{N\times N}$ is covariance matrix. This matrix is positive semidefinite ($v^T\Sigma v \geq 0$ for all $v \in \mathbf{R}^N$), Hermitian and all its eignvalues are positive.
We use Gaussian distributions to make computations easy. The product of two Gaussian distributions is a Gaussian, so the Gaussian exponentail family is closed under multiplication:

$$N(x;a,A)N(x;b,B) = N(x;c,C)N(a;b,A+B) \tag{5.4}$$

where

$$C := (A^{-1}+B^{-1})^{-1}$$

and

$$c := C(A^{-1}a + B^{-1}b)$$

Another property is the closure under marginalization: projections of Gaussians are Gaussian

$$p(z) = N(z;\mu,\Sigma) \Rightarrow p(Az) = N(Az;A\mu,A\Sigma A^T) \tag{5.5}$$

If we have joint distribution over various variables x and y with a mean $(\mu_x,\mu_y)^T$ and a variance $\begin{bmatrix}\Sigma_{xx} & \Sigma_{xy}\\ \Sigma_{yx} & \Sigma_{yy}\end{bmatrix}$ we can select one element of the mean that corresponds to the variable x and the sub matrix which corresponds to it. For example, we can project in one of the coordinate axes, and that's called marginalization, by choosing $A = (1 \quad 0)$

$$\int N[(x,y)^T;(\mu_x;\mu_y)^T, \begin{bmatrix}\Sigma_{xx} & \Sigma_{xy}\\ \Sigma_{yx} & \Sigma_{yy}\end{bmatrix}]dy = N(x;\mu_x,\Sigma_{xx}) \tag{5.6}$$

which is equivalent to the sum rule

$$\int p(x,y)dy = \int p(y|x)p(x)dy = p(x) \tag{5.7}$$

Gaussians are closed under the product rule

$$p(x|y) = \frac{p(x,y)}{p(y)} = N(x;\mu_x + \Sigma_{xy}\Sigma_{yy}^{-1}(y-\mu_y), \Sigma_{xx}\Sigma_{yy}^{-1}\Sigma_{yx}) \tag{5.8}$$

When we sample an n-dimensional normal distribution, we get n numbers which we consider as a set of a point in n-dimensional space. But we can also think of them as the values of a function sampled at endpoints. If we increase n gradually we obtain increased resolution of the function. Points in a finite dimensional space can be sampled from a probability distribution determined by a mean vector and a covariance matrix. Similarly, we can have a probability distribution of functions determined by a mean function and a covariance function. For GPR, the covariance function is determined by a chosen kernel function that describes how much influence one point has on another. This effectively determines the smoothness of the functions in the distribution. Given a set of data points, we can fit a probability distribution to them by choosing the distribution parameters to match the properties of the distribution to the properties of the data. Similarly, given a set of function values, we can fit a probability distribution of functions that closely matches the given function values. Considering the whole fitted distribution of functions, we can determine the mean as well as a confidence interval. This gives us not only a regression function but probabilistic bounds on the prediction as well. Thus, GPR approximates an unknown target function $y = f(x)$ in a probabilistic manner. Given a data set $D_n = \{(x_i, y_i) \mid i = 1,...,n\}$ with n samples, $x_{1:n} = (x_1,...,x_n)$ denote the input and $y_{1:n} = (y_1,...,y_n)$ denote the corresponding output observations. The goal of GPR is to construct a posterior distribution $p(f_{new}|D_n)$ on the function value $f_{new} = f(x_{new})$ corresponding to an unseen target input x_{new}. The prediction is given as a probability distribution rather than a point estimate, quantifying the uncertainty in the target value.
GPR uses Gaussian Process (GP) to discribe the probability distribution on the target function $f(x)$. The function value $f_{1:n} = (f_1,...,f_n)$ corresponding to the input $x_{1:n}$ are treated as random variables, where $f_i := f(x_i)$.
GP is defined as a stochastic process, any number of which is assumed to be jointly Gaussian distributed. GP can fully describe the distribution over an unknown function $f(x)$ by its mean $m(x) = E[f(x)]$ and a kernel function $k(x,x')$ that approximates the covariance $E[(f(x) - m(x))(f(x') - m(x'))]$. The prior on the function values is represented as:

$$(f_{1:n}) = GP(m(.), k(.,.)) \tag{5.9}$$

where $m(.)$ is a mean function (capturing the overall trend in the target function value) and $k(.,.)$ is a kernel function (used to approximate the covariance). We require that $k(.,.)$ be a positive definite kernel.
Suppose we observe the training set D_n wher $f_i = f(x_i)$ is the noise-free observation of the function evaluted at x_i. Given a test set X_* of size $N_* \times D$ wich contains the signals : Roll Amplitude, FuelLpH, FuelLpNm and RudderPos, we want to predict the function outputs f_*

which corresponds to the signal Roll State. If the GP is to predict $f(x)$ for a value of x that it has already seen, the GP will return the answer $f(x)$ with no uncertainty and act as an interpolator of the training data. This only happens if we assume the observations are noiseless.
By the definition of GP, the joint distribution has the following form:

$$\begin{pmatrix} f \\ f_* \end{pmatrix} \sim N\left[\begin{pmatrix} \mu \\ \mu_* \end{pmatrix}, \quad \begin{pmatrix} K & K_* \\ K_*^T & K_{**} \end{pmatrix}\right]$$

where $K = k(X,X)$ is $N \times N$, $K_* = k(X,X_*)$ is $N \times N_*$, and $K_{**} = k(X_*,X_*)$ is $N_* \times N_*$.
By the standard rules for conditional Gaussians, the posterior has the following form

$$p(f_*|X_*,X,f) = N(f_*|\mu_*,\Sigma_*) \tag{5.10}$$

$$\mu_* = \mu(X_*) + K_*^T K^{-1}(f - \mu(X)) \tag{5.11}$$

$$\Sigma_* = K_{**} - K_*^T K^{-1} K_* \tag{5.12}$$

This process is illustrated in Figure 5.8. We apply a moving average filter to remove noise in Figure 5.9. We see that the model interpolates the training data. The rational quadratic kernel is defined as follows:

$$k(x,x') = (1 + \frac{||x - x'||^2}{\nu l^2})^{-\frac{\nu+D}{2}} \tag{5.13}$$

where $\nu > 0$ is the shape parameter, $l > 0$ the scale parameter and D is the dimension of the input space. In GPR, the kernel (covariance) function describes the structure of the target function. Thus, the type of the kernel function $k(x,x')$ used to build a GPR model and its parameters can affect the overall representability of the GPR model and impact the accuracy of the prediction model. However, we choose this model since the training process using Regression Learner in MATLAB, continues until the model achieves a desired level of RMSE (the square root of the variance of the residuals) on the training data.

5.4.2 Ship's displacement

Another example of supervised learning is the regression algorithm. In this case, we want to predict a real or a set of real numbers. Here we discuss two different examples. The first example is the prediction of a ship's displacement (tones) from its dimension. To do so, we use a data set of measurements for length, breadth, and draught of some ships. We suppose also that we know the Deadweight tonnage (DWT) which is the difference between the displacement and the mass of empty vessel (lightweight) at any given draught. The problem here is that we don't know DWT in general. We use stepwise linear regression method to predict the ship's displacement. In this case, instead of using a single predictor variable to make a prediction about a dependent variable, we are using multiple predictor variables to end up with one prediction about a single dependent variable in order to get a more accurate answer. To create the best fit line through the data, we are using the least squares method in which we are finding the smallest sum of the squares of all residuals (differences between the data points and the

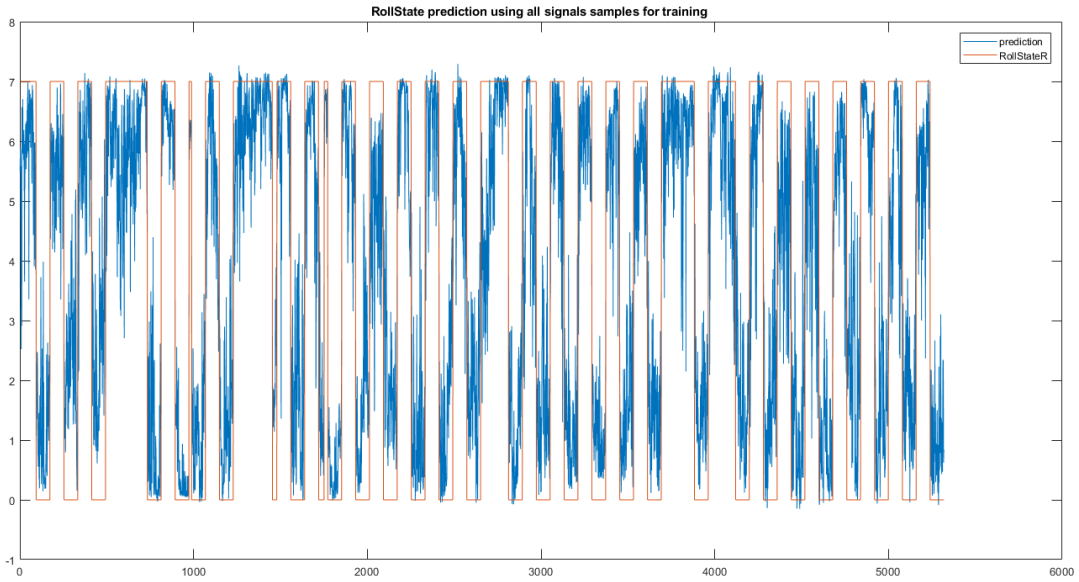


Figure 5.8: RollState signal and prediction.The prediction signal is noisy

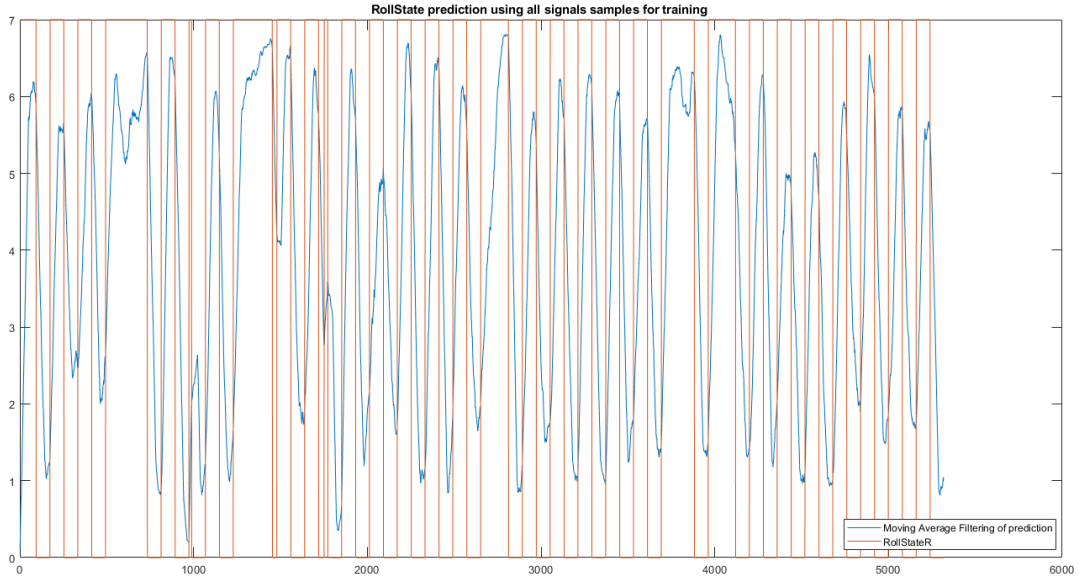


Figure 5.9: RollState signal and prediction.
We apply a moving average filter. we can force the prediction to be a steep function

predicted line), but in a higher dimension.
The equation for multiple regression is given by [13]:

$$y = B_0 + B_1X_{t1} + B_2X_{t2} + B_3X_{t3} + B_4X_{t4} + \varepsilon \tag{5.14}$$

where X_{ti} is the predictor variable and B_i is the corresponding coefficient, and $i = 1,2,3,4$, t is the number of observations (27 in this case) and ε is an error term which represents some amount of uncertainty and it is assumed to be normally distributed with $E[\varepsilon|X = x] = 0$ for all X. Each coefficient B_i describes how y changes when X_{ti} does. The linear regression model expresses the conditional expectation of y given X, as a linear function of X, where X is the set of predictor variables. Thus, by taking the expected value of both sides, the multiple regression equation can be expressed in the equivalent form:

$$E[y|X = x] = B_0 + B_1X_{t1} + B_2X_{t2} + B_3X_{t3} + B_4X_{t4} \tag{5.15}$$

where $B = [\hat{B_0}, \hat{B_1}, \hat{B_2}, \hat{B_3}, \hat{B_4}]^T$ are estimates obtained for the model parameters, which are used to get predicted values of y for each sample observation t:

$$\hat{y_t} = \hat{B_0} + \hat{B_1}X_{t1} + \hat{B_2}X_{t2} + \hat{B_3}X_{t3} + \hat{B_4}X_{t4} \tag{5.16}$$

The fitting problem can be written in the matrix form as:

$$\hat{Y} = XB \tag{5.17}$$

with X is the 27×5 matrix of the 4 predictor vaiables and an 27×1 vector of 1's:$[1\ X_{t1}\ X_{t2}\ X_{t3}\ X_{t4}]$. A vector of residuals e is given by $e = Y - \hat{Y}$. Using least squares method, we minimize the sum of squares of the residuals $\sum_{t=1}^{27} e_t^2$ (see Figure 5.10).

5.4.3 Anti-rolling rules considering rudder position

Ship motions are defined by the six degrees of freedom that a ship can experience. Figure 5.11 shows the axes of a ship and rotations around them. We trigger two types of engine speed change (increase or decrease):

1. A short (pulse) increase or decrease that has the purpose to create a moment change on the ship due to propeller-hull interaction "helicopter propeller effect".
2. A longer speed increase or decrease that has the purpose to create a moment change due to the turning moment of the ship via interaction with the water.

Example: when the ship turns towards port due to a rudder movement to port, then the ship will roll to the starboard. This increase or decrease of the propeller speed must be longer such that the ship is actually changing its speed. If the engine is too weak to accelerate or break the ship in this way, the corresponding rule must be changed to "no action". Thus, Roll Pulse signal depends on both the rolling and the rudder angle. In this example, we define the Roll

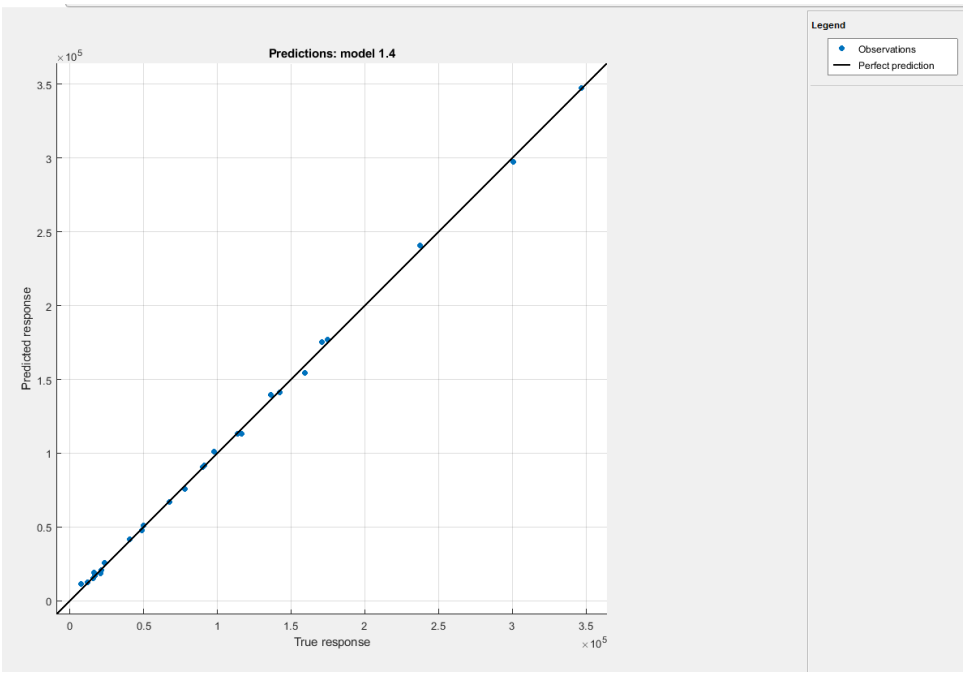


Figure 5.10: Plot Predicted vs. Actual Response.
All the points lie on a diagonal line. The vertical distance from the line to any point is the error of the prediction for that point. A good model has small errors, and so the predictions are scattered near the line.

Pulse signal using rolling and the rudder signals as predictor variables. First we plot all signals to visualize how they're distributed in time, Figure 5.12. We apply the algorithm of the previous subsection to predict Roll Pulse signal. Figure 5.13 shows Roll Pulse signal and its prediction.

5.5 Identification of coherent groups of signals

As we mentioned earlier, it is not a good approach to use too many predictor variables to estimate or to predict another one. In order to identify related signals, we use mutual information (MI) computation. In probability and information theory, the MI of two signals is a measure of the mutual dependence between the two signals. We could compute the correlation coefficient, but this is a very limited measure of dependence since for example a signal Z may be not correlated with two others X and Y, but is highly correlated with some function $f(X,Y)$. The MI between three signals X , Y and Z is equal to the sum of the MI between X and Y, plus the MI between Z and the combined signal (XY)

$$MI(X,Y,Z) = MI(X,Y) + MI((X,Y),Z) \tag{5.18}$$

Thus, to predict which kinds of data are likely and which unlikely, we use MI as a more general approach, so that we can determine -for two signals X and Y- how similar the joint distribution

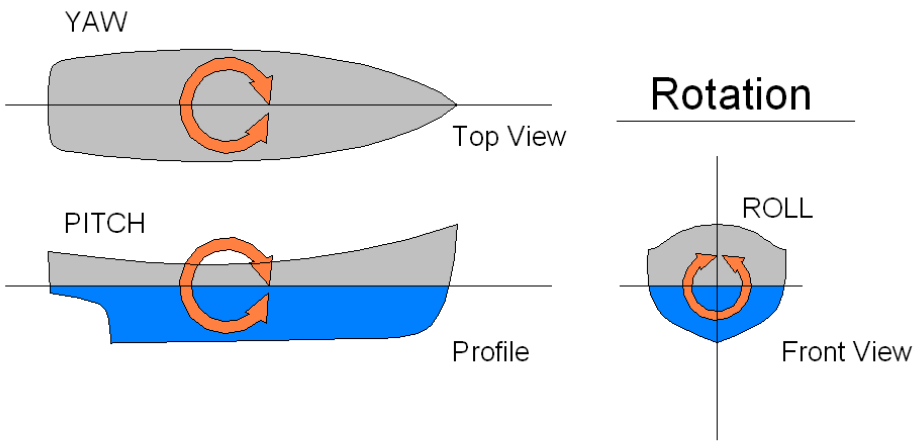


Figure 5.11: Axes of a ship and rotations around them.
From Wikimedia Commons, the free media repository

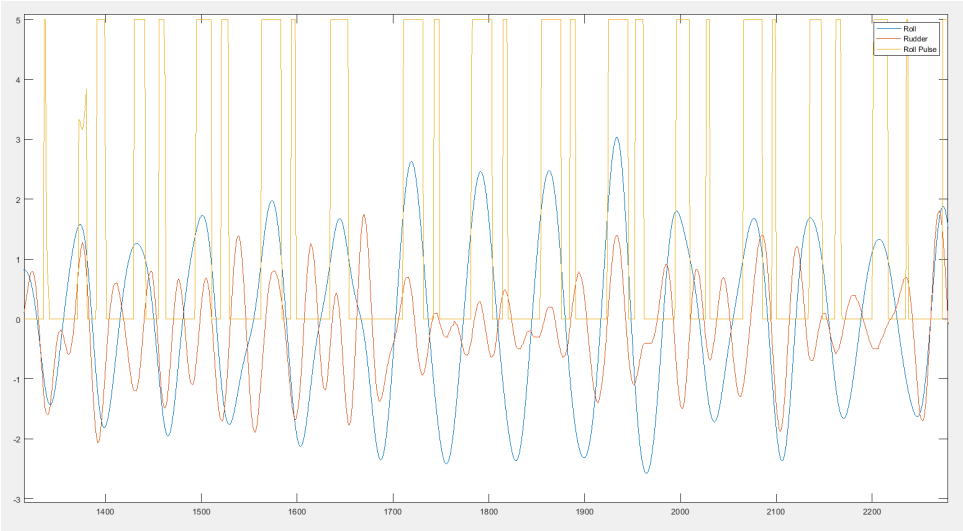


Figure 5.12: Roll, Rudder and RollPulse signals.

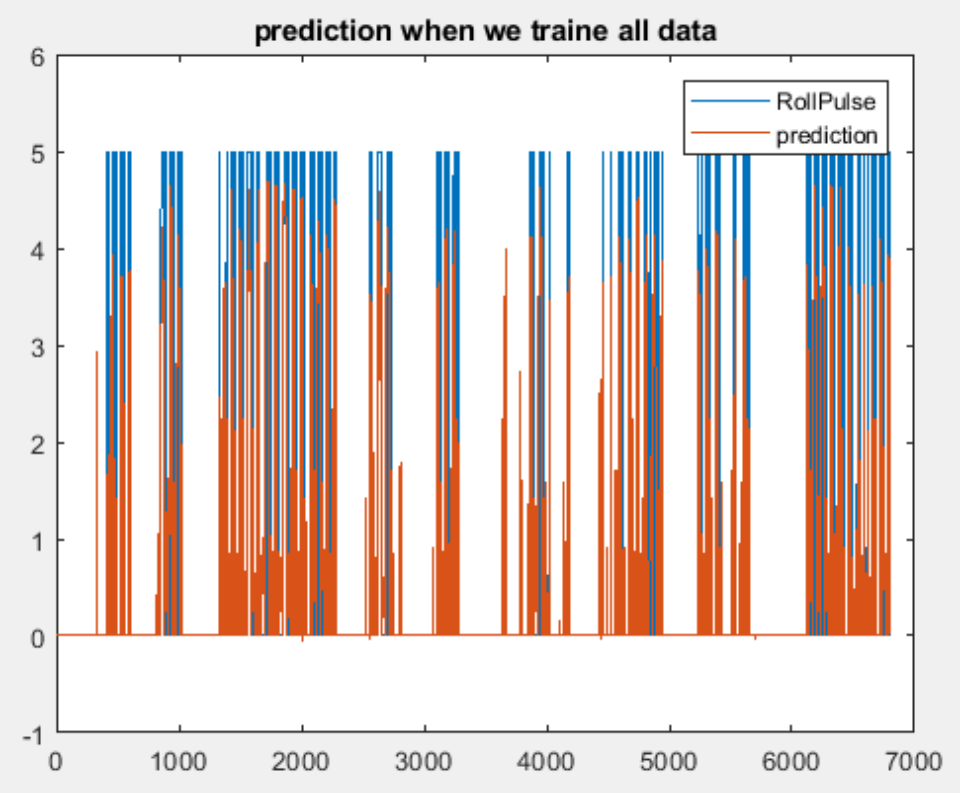


Figure 5.13: Roll Pulse signal and prediction.
In this case, we train all data and we plot the Roll Pulse signal and its prediction: most of the time, the algorithm can predict the signal. We can force our prediction to be a step function by saying that the value will be 5 if we reach a certain threshold, and zero otherwise.

$p(X,Y)$ is to the factored distribution $p(X)p(Y)$. This is given by [7]

$$MI(X,Y) = \sum_x \sum_y p(x,y) log \frac{p(x,y)}{p(x)p(y)} \tag{5.19}$$

The normalized mutual information is given by [7]

$$MI_{normalized}(X,Y) = \frac{MI(X,Y)}{min(H(X),H(Y))} \tag{5.20}$$

with H refers to the entropy (the uncertainty of a signal) which is defined by

$$H(X) = -\sum_{n=1}^{N} p(X=n) log p(X=n) \tag{5.21}$$

with X is a discrete signal of length N and p is its distribution. Indeed, for two signals X and Y, we write $p_i(X) = prob(X = x_i)$, $p_i(Y) = prob(Y = y_i)$ and $p_{ij} = prob(X = x_i, Y = y_i)$ for the marginal and joint distribution. The entropie of the joint distribution (X,Y) is given by

$$H(X,Y) = -\sum_{i,j} p_{ij} log p_{ij} \tag{5.22}$$

and conditional entropies is defined as

$$H(X|Y) = H(X,Y) - H(Y) = -\sum_{i,j} p_{ij} log p_{i/j} \tag{5.23}$$

The MI between X and Y is defined as

$$MI(X,Y) = H(X) + H(Y) - H(X,Y) = \sum_{i,j} p_{ij} log \frac{p_{ij}}{p_i(X)} \tag{5.24}$$

which is zero when X and Y are strictly independent. For N signals X_1, X_2... X_N, the MI is given by

$$I(X_1,...,X_N) = \sum_{k=1}^{N} H(x_k) - H(X_1,X_2,...,X_N). \tag{5.25}$$

For three signals[14], we have

$$I(X,Y,Z) = H(X) + H(Y) + H(Z) - H(X,Y,Z) \tag{5.26}$$

$$= \sum_{i,j,k} p_{ijk} log \frac{p_{ijk}}{p_i(X) p_j(Y) p_k(Z)} \tag{5.27}$$

$$= \sum_{i,j,k} p_{ijk} [log \frac{p_{ij}(XY)}{p_i(X) p_j(Y)} + log \frac{p_{ijk}}{p_{ij}(XY) p_k(Z)}] \tag{5.28}$$

$$= MI(X,Y) + MI((X,Y),Z) \tag{5.29}$$

The definition of independent subsets can then be performed by an unsupervised classification method so that MI ensure that the signals are related in each group. We use a clustering algorithm to classify our signals into its coherent groups.

In hierarchical clustering methods, we start with each data point as its own cluster and then combine two nearest clusters into a larger cluster and repeat the process. In general, to build such hierarchical clustering algorithm, we measure cluster distances by distances of centroids of clusters using Euclidean distance. In the case of Non-Euclidean distance, we can't average points and create a centroid, so there is no concept of average and therefore we can not continue using the centroid to represent a cluster. Instead, we can represent a cluster of many points using clustroid, which is an existing data point that is closest to the other points in the cluster. To stop clustering and produce an output (otherwise we end up with one large cluster), we can choose a number k and stop when we have k clusters. This is useful when we know that the data naturally falls into k classes. If we don't know k, we stop when the next merge would create a bad cluster. We can measure the goodness of a cluster using "cohesion". By cohesion it is meant a value which falls below a certain level then a bad cluster is created . Cohesion is defined by a diameter of the merged cluster which is the maximum distance between two points in the cluster. It can be defined by a radius, which is the maximum distance of the point from clustroid. We can also use the notion of density to stop clustering. To do so, we divide the number of points in the cluster by the diameter or radius of the cluster. In each case, we predefine a threshold either for diameter, radius or density and stop when the next merge would achieve that threshold.

We use MI as proximity measure to obtain independent groups of signals so that each signal X_i belongs to a group of signals G_j is sharing information with another one. To divide the set of all signals into homogeneous groups, we use cluster analysis as an unsupervised process. Here we introduce the scheme for clustering N signals [14]:

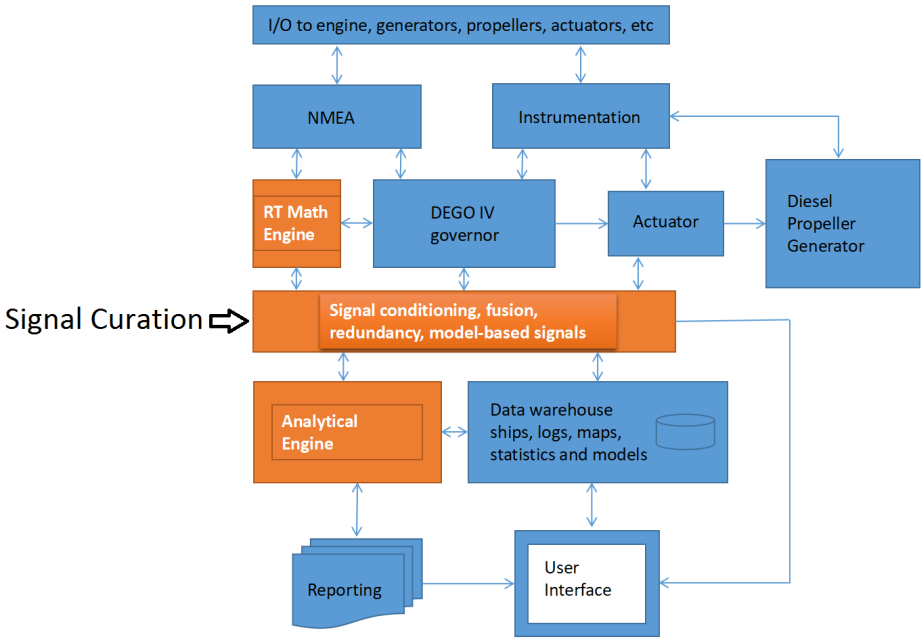
1- Compute a proximity matrix based on the pairwise mutual information.
2- Assign N clusters such that each cluster contains one single signal.
3- Find the two closest clusters i and j.
4- Combine the two clusters i and j into one new cluster (ij).
5- Delete the lines/columns with indices i and j from the proximity matrix and add one line/column containing the proximities between cluster (ij) and all other clusters.
6- Go to the third step if the number of clusters is still bigger than 2. Otherwise, joint the two clusters and stop.

Chapter 6

Conclusion

ML allows computer systems to learn directly from examples, data, and experience. Through enabling computers to perform specific tasks intelligently, ML can carry out complex processes by learning from data, rather than following pre-programmed rules. The behavior of the system depends on the quality of each input. However, to study separately several subsets of variables, we may identify coherent groups of variables using MI as proximity measure. The definition of independent subsets can then be performed by unsupervised classification methods. The most natural idea to detect the abnormal behavior of signals while keeping flexibility is to exploit regression models by predicting a variable using the variables belonging to the same group, and this is the main objective of the Thesis.

The general organization of QTAGG's system is shown in the following Figure. The emphasis in this thesis is shown by the second orange block.



Chapter 7

Summary of reflection of objectives in the Thesis

Following the result of this work, the system is ready to be implemented. The system integration is shown in architecture figure (see previous page). The system works as follows:
Step 1: Accessing and exploring in real-time signals from sensors
Step 2: Preprocessing signals by detecting and correcting outliers or missing values
Step 3: Extracting features and create groups of related signals
Step 4: Running predictive model identification methods, train and compare multiple models, optimize parameters and validate models performance to ensure robustness
Step 5: The final step is sharing and deploying trained models by integrating them into analytic pipelines. Models may run on embedded systems or shared as standalone application services or could be even deployed on the cloud

One contribution of the thesis is to provide a broad and relevant knowledge of how mathematics is used in signal-intensive applications. This knowledge is in high demand in industry since most technologies depend on mathematical methods and calculations. Until now, these calculations have done mostly manually and not by automated means. The approach pursued in this thesis shows what type of knowledge could be used in developing these new technology. It shows that improving the skills in Engineering Mathematics implies advanced methods in statistics, machine learning, signal processing, computer science and optimization. All these are required in applications such as electrical power industry, environmental engineering, robotics and control engineering.

The goal of this project was to find the mathematical principles of a system which learns automatically the essential statistical and analytical properties of signals by using ML technique in order to detect and correct certain classes of faults in real time. The main idea to achieve such objective is the application of ML techniques to find optimal analysis algorithms for signals.

A mathematical challenge in ML is that ML methods are not precise with small data sets but the precision comes asymptotically with very large datasets. A future challenge would

be to compute the minimum size of the data set that achieve a given precision. Theoretical results in this field are available but are harder to automate and this was beyond the scope of this thesis. We solved this problem - since we had to work with one example of data from an ocean going vessel - by splitting the data into subintervals and extracting features from each subinterval (K-fold cross validation method). Another aspect also used was to add a number of random values to each observation. However, the analysis will be more precise if many logs are available at faster sampling time (e.g. 250 milliseconds instead of 3 minutes).

It was a challenge for me to find the relevant literature in autonomous mathematical methods for signal analysis and correction since these methods appear to be falling between many different specific research fields. However, thanks to advices from my supervisors at QTAGG R&D AB, MDH and thanks to the instructions from my professors at my home Galilee Institute at University Paris 13 in France, I succeeded to analyze, assess and deal with such complex issues and situations and finish the thesis in time.

Bibliography

[1] EiC Vijay K. Madisetti, CRC Press, Taylor and Francis Group, 2010.
The Digital Signal Processing Handbook, Second Edition.

[2] Signal Processing Toolbox, Reference, MathWorks, R2018, 2018.

[3] IFAC: Working group in FDI technology,
`https://tc.ifac-control.org/6/4/working-groups/industrial-application-of-advanced-fdi-ftc-technology`

[4] Patton, R.J., Frank, P.M., Clark, N.R. (eds.), Springer, 2000.
Issues of Fault Diagnosis for Dynamical Systems

[5] Julier, S., Uhlman, J.K. General Decentralized Data Fusion with Covariance Intersection, in the Handbook of Multisensor Data Fusion, Theory and Practice, Second Ed., CRC Press Taylor and Francis, 2009, pp. 319-342

[6] Mahler, R., Random Set Theory for Multisource-Multitarget Information Fusion, Theory and Practice, Second Ed., CRC Press Taylor and Francis, 2009, pp. 369-407.

[7] Seichepine, Ricordeau, Lacaille - 2011, Snecma, Rond-Point René Ravaud, Réau, 77550 Moissy-Cramayel cedex, France
Data mining of flight measurements.

[8] George Fodor, Q-TAGG R&D AB, Västerås, Sweden
Qtagg's Approach to Fuel Savings Measurement, internal document.

[9] Maths Works, Hampel Filter
`https://se.mathworks.com/help/dsp/ref/hampelfilter.html`

[10] Kalman filter. From Wikipedia, the free encyclopedia,
`https://en.wikipedia.org/wiki/Kalman filter`

[11] The Kalman Filter, Tony Lacey
`http://web.mit.edu/kirtley/kirtley/binlustuff/literature/control/Kalman%20filter.pdf`

[12] Math Works,
`Human Activity Classification based on Smartphone Sensor Signals`

[13] Ching-Kang Ing and Tze Leung LAI-Academia Sinica and Stanford University,
A stepwise regression method and consistent model selection for high-dimensional sparse linear models.

[14] Kraskov, A., Grassberger, P., MIC:Mutual Information based hierarchical Clustering, Cornell University Library, arXiv.org: 0809.1605, Submitted 9 Sep 2008.

Matlab Code

```
% classification of signals : STW & RollPitch signals using SVM
stw = [9.1435    56.9606  24.4    7.5472 % max, mean; var & std of STW
rp =  [0.2278    0.0178   0.9      0.1336]; % max, mean; var & std of Roll Pitch
data_stw = observations(stw);  % Add 100 randn to each feature to end up with 100 observations
data_rp = observations(rp);
data = [data_stw;data_rp];
% normalize data
for k=1:size(data,1)
    data_nor = data/sum(data(k,:));
end
label_stw = zeros(100,1);
label_rp = ones(100,1);
label=[label_stw;label_rp];
data_table=[data_nor label];
% train data
[trainedClassifier, validationAccuracy] = trainClassifier(data_table);
% New signal
NewSignal = V ;
y=f_feature(NewSignal); % max, mean; var & std
yfit = trainedClassifier.predictFcn(y); % return prediction label , 0 for STW & 1 for RP
if (yfit == 0)
    disp('This signal is STW')
else
    disp('This signal is RP')
end
```

```
function y=feature(NewSignal)
y = [mean(NewSignal) var(NewSignal) max(NewSignal) std(NewSignal)];
end
```

```
function [M_data]=observations(v)

N = length(v);
M_data=[];
% 100 continuous measurement for each observation
```

```
for i=1:N
Data_M = v(i) +abs( randn(100,1));
M_data =[M_data   Data_M];
end
end
```

```
% classification of signals : this code predict RollState
% signal from data = [RollAmpR FuelLpHR FuelLpNmR RudderPosR]
% test : Rgression => use all data to train the  algorithm
% Access and Explore Data from Kampala 2017-12-01 to
   2018-01-22
data = [RollAmpR FuelLpHR FuelLpNmR RudderPosR RollStateR];
my_data = data;
Signal_to_predict = RollStateR; % from a new log
% Preprocess Data & Develop Predictive Models
NumberOfSignals = size(data,2);
observations=[];
% first we split data to end up with many observations
T = 100 ;  % Number Of Observations
mini_data=SplitData(data,T);
for l = 1 : T
    data = mini_data{l};
    observations = [observations ; data];
    % now we extract features
    % for k = 1 : NumberOfSignals
    % f = xtract_features(data(:,k));
    % F = [F f];
end
trainingData = observations;
%trainingData=data;
[trainedModel, validationRMSE] = Rational_Quadratic_GPR(
   trainingData);
New_signals = [RollAmpR FuelLpHR FuelLpNmR RudderPosR];
prediction = trainedModel.predictFcn(New_signals);
figure(1)
plot(prediction)
hold on
plot(RollStateR)
legend('prediction','RollStateR')
title('RollState prediction using all signals samples for
   training ')
figure(2)
plot(MovingAverageFiltering( prediction ))
hold on
```

```
plot(RollStateR)
legend('Moving Average Filtering of prediction ','RollStateR')
title('RollState prediction using all signals samples for
    training ')
```

```
% classification of signals : this code predict RollState
% signal from data = [RollAmpR FuelLpHR FuelLpNmR RudderPosR]
% test : Rgression => use all data to train the  algorithm
% Access and Explore Data from Kampala 2017-12-01 to
    2018-01-22
data = [RollAmpR FuelLpHR FuelLpNmR RudderPosR RollStateR];
my_data = data;
Signal_to_predict = RollStateR; % from a new log
% Preprocess Data & Develop Predictive Models
NumberOfSignals = size(data,2);
observations=[];
% first we split data to end up with many observations
T = 100 ;  % Number Of Observations
mini_data=SplitData(data,T);
for l = 1 : T
     data = mini_data{l};
     observations = [observations ; data];
     % now we extract features
     % for k = 1 : NumberOfSignals
     % f = xtract_features(data(:,k));
     % F = [F f];
end
trainingData = observations;
%trainingData=data;
[trainedModel, validationRMSE] = Rational_Quadratic_GPR(
    trainingData);
New_signals = [RollAmpR FuelLpHR FuelLpNmR RudderPosR];
prediction = trainedModel.predictFcn(New_signals);
figure(1)
plot(prediction)
hold on
plot(RollStateR)
legend('prediction','RollStateR')
title('RollState prediction using all signals samples for
    training ')
figure(2)
plot(MovingAverageFiltering( prediction ))
hold on
plot(RollStateR)
```

```
37 legend('Moving Average Filtering of prediction ','RollStateR')
38 title('RollState prediction using all signals samples for
       training ')
```