

**MINING AND FUSING DATA FOR OCEAN TURBINE CONDITION
MONITORING**

by

Janell A. Duhaney

A Dissertation Submitted to the Faculty of
The College of Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

Florida Atlantic University

Boca Raton, FL

December 2012

Copyright by Janell A. Duhaney 2012

MINING AND FUSING DATA FOR OCEAN TURBINE CONDITION MONITORING

by

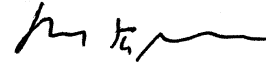
Janell A. Duhaney

This dissertation was prepared under the direction of the candidate's dissertation advisor, Dr. Taghi M. Khoshgoftaar, Department of Computer and Electrical Engineering and Computer Science, and has been approved by the members of her supervisory committee. It was submitted to the faculty of the College of Engineering and Computer Science and was accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

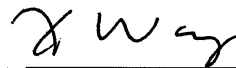
SUPERVISORY COMMITTEE:



Taghi M. Khoshgoftaar, Ph.D.
Dissertation Advisor



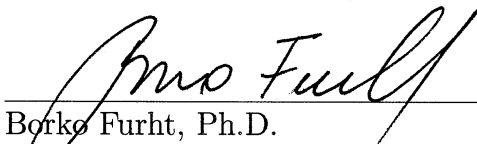
Martin K. Solomon, Ph.D.



Xin Wang, Ph.D.



Bassem Alhalabi, Ph.D.



Borko Furht, Ph.D.
Chair, Department of Computer and Electrical
Engineering and Computer Science



Mohammad Ilyas, Ph.D.
Interim Dean, College of Engineering and
Computer Science



Barry T. Rosson, Ph.D.
Dean, Graduate College



Date

ACKNOWLEDGEMENTS

Foremost, I thank God for giving me the strength and perseverance to complete this research. I can do all things through Him who strengthens me.

I would like to express my deepest gratitude to my advisor, Dr. Taghi M. Khoshgoftaar, for his guidance and encouragement throughout my doctoral studies at Florida Atlantic University. His many years of research experience, excellent advice and never tiring attitude made completion of this degree possible. Thanks also to Dr. Martin K. Solomon, Dr. Xin Wang, and Dr. Bassem Alhalabi for serving on my dissertation committee.

I wish to thank the Southeast National Marine Renewable Energy Center (SNMREC) for supporting and funding this research. I also acknowledge Randall Wald, Dr. Amri Napolitano, and Dr. John C. Sloan, who I had the pleasure of working alongside in the Data Mining and Machine Learning Lab, as well as the graduate students, faculty and staff in the Department of Ocean and Mechanical Engineering at the Dania Beach campus who also worked on the SNMREC ocean turbine project.

ABSTRACT

Author: Janell A. Duhaney
Title: Mining and Fusing Data for Ocean Turbine Condition Monitoring
Institution: Florida Atlantic University
Dissertation Advisor: Dr. Taghi M. Khoshgoftaar
Degree: Doctor of Philosophy
Year: 2012

An ocean turbine extracts the kinetic energy from ocean currents to generate electricity. Machine Condition Monitoring (MCM) / Prognostic Health Monitoring (PHM) systems allow for self-checking and automated fault detection, and are integral in the construction of a highly reliable ocean turbine. MCM/PHM systems enable real time health assessment, prognostics and advisory generation by interpreting data from sensors installed on the machine being monitored.

To effectively utilize sensor readings for determining the health of individual components, macro-components and the overall system, these measurements must somehow be combined or integrated to form a holistic picture. The process used to perform this combination is called data fusion. Data mining and machine learning techniques allow for the analysis of these sensor signals, any maintenance history and other available information (like expert knowledge) to automate decision making and other such processes within MCM/PHM systems.

Our research investigates the feasibility of various data mining, machine learn-

ing and data fusion techniques for an MCM/PHM system. Studies conducted on experimental data aim to reveal the optimal approach for fusing and interpreting sensor data. Also considered in these studies is the possibility of imperfect data and other challenges that could negatively affect the efficiency of our techniques. Specifically, we assess the robustness of our techniques to changing environmental conditions, class imbalance (*i.e.*, the relative lack of fault data as compared to data collected during normal operation that will be available to construct state detection models) and data incompleteness (*e.g.* missing values in the data).

This dissertation proposes an MCM/PHM software architecture employing those techniques which were determined from these experiments to be ideal for this application. Our work also offers a data fusion framework applicable to ocean machinery MCM/PHM. Finally, it presents a software tool for monitoring ocean turbines and other submerged vessels, implemented according to industry standards.

DEDICATION

I dedicate this degree to my parents, Mr. and Mrs. Lansvill Duhaney, my brothers, Lansville and Kevon, and closest friends Tajheim, Monique and Ingrid, who kept me grounded over the past years. I also dedicate my efforts to my late aunt, Carmen Smith, whose love for God, life, family and education will never be forgotten.

MINING AND FUSING DATA FOR OCEAN TURBINE CONDITION MONITORING

List of Tables	xiii
List of Figures	xiv
1 Introduction	1
1.1 Motivation	2
1.2 Contributions	6
1.3 Dissertation Structure	7
2 Background	9
2.1 Components of the Dynamometer	11
2.2 MCM/PHM	14
2.2.1 Data Acquisition (DA)	15
2.2.2 Data Manipulation (DM)	15
2.2.3 State Detection (SD)	16
2.2.4 Health Assessment (HA)	16
2.2.5 Prognostics Assessment (PA)	16
2.2.6 Advisory Generation (AG)	16
2.3 Overall Architecture	16
3 Related Work	20
3.1 Data Mining & Machine Learning	21

3.1.1	Data Stream Mining	22
3.1.2	Data Stream Management	27
3.2	Data Fusion	30
3.2.1	Uncertainty	32
3.2.2	Imprecision	32
3.2.3	Vagueness	33
3.2.4	Incompleteness	33
3.2.5	Inconsistencies	34
3.2.6	Correlation	34
3.3	Data Fusion Techniques	35
3.3.1	Bayesian Theory	37
3.3.2	Possibility Theory	38
3.3.3	Fuzzy Set Theory	39
3.3.4	Evidence Theory	40
3.3.5	Other	42
3.4	Conclusion	44
4	Methodology	46
4.1	Data Acquisition	46
4.2	Data Manipulation/Transformation & Feature Extraction	48
4.2.1	Streaming Wavelet Transform	50
4.2.2	Short Time Wavelet Transform	52
4.3	Fusion Techniques	53
4.3.1	Data-Level Fusion	53
4.3.2	Feature level fusion	54
4.3.3	Decision level fusion	54
4.4	Classifiers	55

4.5	Feature Selection	59
4.6	Data Imputation	61
4.7	Performance Measures	61
4.8	Performance Evaluation	63
5	Data and Knowledge Fusion Framework for MCM/PHM in Inaccessible Ocean Systems	65
5.1	Introduction	65
5.2	Chapter Contributions	65
5.3	Related Work	66
5.3.1	Data Fusion	67
5.4	Framework	69
5.5	Case Study: Ocean Turbine MCM/PHM	70
5.5.1	Applying the Framework	71
5.5.2	Barrier Synchronization	72
5.5.3	Inter-Sensor Data Fusion	75
5.5.4	Intra-Component Data Fusion	76
5.5.5	Inter-Component Data Fusion	78
5.6	Experimental Setup	80
5.6.1	Results	84
5.7	Chapter Summary	86
6	Preliminary Analysis of Data Mining & Sensor Fusion Techniques on Fan Data	88
6.1	Introduction	88
6.1.1	Contributions	89
6.2	Experimental Setup	90
6.2.1	Fan Experiment 1	91

6.2.2	Fan Experiment 2	91
6.2.3	Pre-Processing and Fusion	92
6.3	Empirical Results	95
6.3.1	Results for Fan Experiment 1	96
6.3.2	Results for Fan Experiment 2	97
6.4	Conclusion	98
7	State Detection From Imperfect Data	100
7.1	Introduction	100
7.2	Contributions	101
7.3	Case Study 1 – Missing Data	102
7.3.1	Experimental Design	104
7.3.2	Pre-Processing	105
7.3.3	Classifiers	108
7.3.4	Classification	109
7.3.5	Performance Measures	110
7.3.6	Results	111
7.3.7	Data Imputation Results	114
7.4	Case Study 2 – Feature Selection	116
7.4.1	Feature Selection on SWT Data	117
7.4.2	Results	118
7.4.3	Feature Selection on STWTB Data	119
7.4.4	Results	121
7.5	Conclusion	124
8	MCM/PHM In The Presence of Class Imbalance	127
8.1	Introduction	127

8.2	Contributions	129
8.3	Background	130
8.3.1	Data level approaches	131
8.3.2	Algorithmic level approaches	132
8.4	Related Work in Reliability Analysis Applications	133
8.5	Methodology	135
8.5.1	Learners	135
8.5.2	Feature Level Fusion	136
8.5.3	Decision Level Fusion	136
8.6	Experimental Setup	137
8.6.1	Data Acquisition	137
8.6.2	Data Transformation	138
8.6.3	Class Distributions	139
8.6.4	Performance Measure	140
8.7	Results & Analysis	141
8.7.1	How does class imbalance in these datasets affect the results using the specified waveform analysis and machine learning techniques?	142
8.7.2	Which learners are affected most and least by class imbalance in these datasets?	145
8.7.3	Which data fusion approach is better suited for this problem?	146
8.7.4	Which learners perform best regardless of the speed at which the training data was collected?	149
8.7.5	What is the optimal combination of data fusion type (feature level or decision level) and learner?	149
8.7.6	Analysis of Results	150
8.8	Chapter Summary	153

9	Condition Monitoring Software System (CMSS)	156
9.1	Introduction	156
9.2	CMSS	158
9.2.1	Requirements	158
9.2.2	Architecture	159
9.2.3	Implementation	163
9.3	Conclusion	166
10	Conclusion and Future Work	167
10.1	Conclusions	167
10.2	Future Work	170
	Bibliography	172

LIST OF TABLES

7.1	Summary of Feature Selection Techniques Per Classifier	120
7.2	Classification Accuracies per Feature Selection Technique, Learner and # of Features	122
8.1	Training Set Distributions	139
8.2	ANOVA – 25RPM	151
8.3	ANOVA Results – 50RPM	151

LIST OF FIGURES

2.1	Closeup of nacelle	9
2.2	Turbine and its moorings	10
2.3	Picture of the Dynamometer	10
2.4	Dynamometer Diagram Showing Sensor Locations	12
2.5	High Level MCM/PHM Architecture Showing Data Flow	19
3.1	Data Level Fusion	36
3.2	Feature Level Fusion	36
3.3	Decision Level Fusion	37
4.1	Wavelet Transform Filter Bank (Credit: Wikimedia Foundation, Inc.)	51
5.1	JDL Process Model for Data Fusion	66
5.2	Data Fusion and MCM Model	68
5.3	Barrier synchronization of timed data streams	73
5.4	Intra-component fusion diagram	76
5.5	Inter-component fusion diagram	78
5.6	Results	83
6.1	Fan Experiment 1 - AUC per Learner and Fusion Technique	96
6.2	Fan Experiment 2 - AUC per Learner and Fusion Technique	98
7.1	Example of Applying Haar Wavelet Transform Showing Missing Values	106
7.2	Sample Output of Wavelet Transform	108

7.3	Average Classification Results Per Learner	112
7.4	Average accuracy by learner before and after imputing data for Experiment A - BL vs HH and BL vs SH	114
7.5	Average accuracy by learner before and after imputing data for Experiment B - BL vs SH where BL and SH recorded 3 times at different speeds	115
7.6	Average accuracy by learner before and after imputing data for Experiment C - BL vs 40% Load	116
7.7	AUC per Learner and Feature Selection Technique	126
8.1	Classification Results (AUC) For All Learners With 25RPM Training Set	143
8.2	Classification Results (AUC) For All Learners With 50RPM Training Set	144
8.3	Change in AUC For All Learners On Balanced and Imbalanced Fused Data	146
8.4	Error Rates	148
8.5	Optional caption for list of figures	155
9.1	Data Flow through MCM/PHM system	160

Chapter 1

Introduction

Finding clean, renewable alternative energy sources has become a worldwide initiative due to the increase in world energy consumption over the past years, the growing concerns for waning fossil fuel reserves and the environmental impact associated with the use of these fossil fuels. One such alternative involves using turbines to extract the natural energy from the steady unidirectional flow of ocean currents such as the Gulf Stream. It is estimated, for example, that if only 0.1% of the potential energy in the Gulf Stream is captured, it would satisfy 35% of the energy demand in Florida, U.S.A. [104].

Currents in the Gulf Stream are driven primarily by wind stress and equatorial solar heating. The opposing forces of the trade winds (blowing westward) and the westerlies (blowing eastwards) apply a stress to the subtropical ocean surface; this, together with a process known as western intensification, makes the resulting Gulf Stream a strong ocean current and an excellent candidate for hydrokinetic energy production (the generation of renewable electricity from the kinetic energy of a body of water). The Gulf Stream itself is roughly 100 kilometres in width and between 800 and 1,200 metres in depth. Due to these depths, fixed structure installations are highly infeasible making floating tethered structures (such as the turbine prototype design described in later chapters) the optimal design choice.

1.1 MOTIVATION

To harness underwater ocean currents, ocean turbines operate unattended below the ocean's surface, which in itself creates unique reliability concerns [4]. Reliability is a particularly important issue for several reasons. Firstly, retrieval of the turbine for manual inspections is accompanied by high expeditionary costs. Also, these turbines are subject to harsh and unpredictable environmental conditions which may induce additional faults. Finally, ocean turbines are required to meet output and uptime requirements. Some of the reliability issues related to development, maintenance and deployment of an ocean turbine are:

1. Bio-Fouling – While submerged, the ocean turbine is susceptible to biological fouling, or bio-fouling, which is the gradual but undesired accumulation of animals and plants on the turbine. Sensors on a buoyancy-driven underwater glider developed in 2003 as a part of the UCSD Spray Project to observe oceanographic features stopped functioning within a mere four weeks of deployment in the Monterey Canyon due to bio-fouling [132].
2. Corrosion – The salinity of the ocean water behaves as a corrosive agent for parts of the turbine, with cabling being an easy target. Destruction of cabling leads to a loss of communication between the turbine and any components on the ocean surface, while corrosion of the nacelle could eventually cause a breach. Also, because these turbines are tethered, corroded cabling could also result in the detachment and loss of the turbine itself.
3. Turbidity – Oceanic wildlife or debris could impact or obstruct the turbine. Also, larger objects could tilt the turbine or cause it to lean on its mooring line.
4. Cavitation – Cavitation, or the formation of bubbles, is also possible at any

point on the structure where the local pressure level on that section is reduced to (or below) the level of saturated vapour pressure of the ambient water. This phenomenon has been known to cause significant wear, erosion of the structure, shaft vibration and possible performance degradation [155].

Reliable and timely detection of problems is a must to avoid damage to these expensive machines. However, frequent manual inspections are infeasible due to high expeditionary costs to access the machines, and thus problems which occur between inspection intervals can go undetected until the next maintenance visit. The complexity of the turbine along with the previously named factors demand an automated monitoring and self-checking process. One such automated solution is a machine condition monitoring / prognostics health monitoring (MCM/PHM) system.

MCM/PHM systems provide automated monitoring and other capabilities, such as predicting the likelihood of future faults and estimating time to failure under given operational conditions. Another added benefit of using an MCM/PHM system is that it allows for predictive maintenance of a machine (where maintenance tasks are performed as needed) versus routine (or time-based) preventative maintenance (where maintenance activities are scheduled at predetermined intervals). In a preventative maintenance scheme, one challenge is ascertaining how frequently such maintenance activities should be performed. Too frequent visits could end up wasting money and resources while infrequent visits run the risk of faults developing between scheduled visits which may damage the machine and cause unnecessary downtime. By performing automated monitoring of the ocean turbine, potential faults can be detected and identified at an early stage allowing for quick remediation, such as self adjustment or shutdown, to minimize damage to the turbine and in turn, reduce downtime due to failure. In this way, an MCM/PHM system instills high assurance of turbine

reliability and increased productivity.

MCM/PHM systems employ a network of sensors to record data about the behaviour and operational environment of the systems they monitor. Such systems record and interpret data from sensors attached to different components of the machine to detect faults, predict future failures and generate advisories for improving the lifetime of the machine. This automated self-checking can be made possible through the use of vibration analysis, data mining and data fusion techniques [39].

One physical phenomenon that tells a great deal about the state of a machine is its vibration. Vibration signals contain a lot of useful information, but in their raw form, these signals cannot be easily interpreted. Raw vibration data are a time-series of amplitude/magnitude/displacement data, which take the shape of waves when the amplitude values are graphed on the y-axis against time on the x-axis. Vibration analysis is the field of study dedicated to understanding these waveforms and to detecting and analyzing patterns in the signals.

Data mining and machine learning, which collectively refer to techniques for inferring knowledge from raw data by analyzing patterns, provide an avenue for automated interpretation of the sensor data and problem classification. In this setting, data mining and machine learning techniques can help automate fault detection, identify failure states and extract patterns in operational state and environment from the massive amount of data generated from the sensors. These techniques can also predict life and future health of the machine.

While most mechanical defects can be determined through analysis of the vibration data generated from sensors known as accelerometers, the data gathered from the remaining sensors are necessary to fully assess the state, life expectancy and potential failure modes of the machine [105]. The data from these various sensor types are usually generated with differing frequencies and with dissimilar representations. Data

fusion techniques are needed to combine data from multiple sources to get a complete, more accurate picture. These techniques are especially useful in an MCM/PHM system for combining sensor data at different stages of the monitoring process to produce more accurate and complete results.

The challenge behind constructing an MCM/PHM system is determining the optimal approach to integrating hardware and software components and manipulating, processing, analyzing, recording, combining and interpreting the sensor data (along with historical data, expert information and data from other sources) so as to perform the required diagnostic and prognostic functions. Fortunately, the International Organization for Standardization (ISO) devised a template for doing just this. Designing an MCM/PHM system that satisfies the ISO standards (specifically ISO-13374 [74]) offers many benefits including the production time and dollars saved by integrating existing solutions which also satisfy the ISO-13374 standard.

There are several challenges involved in building a reliable MCM/PHM system for ocean turbines. These include:

1. Determining when and how to combine data and information within the system.
Data fusion, the process of combining information from multiple sources to gain a unified perspective, is a cross cutting concern in MCM/PHM as it is needed at various points within the system to integrate data and information which may be in different formats, may arrive late or out of sequence or may be conflicting, incomplete or inaccurate [41].
2. Data characteristic of faulty or abnormal states may be lacking or unavailable.
Machine learning algorithms rely on a sample of data representing the various classes/categories/states of interest to build classifiers which can distinguish each state. Class imbalance – the relative lack of examples in one or more classes

compared to the others – is an issue plaguing many real world applications as classifiers usually tend to label all examples (or instances) as being of one of the majority classes to maximize overall accuracy (percentage of correctly identified examples) [72]. This means that rare events or faults, which are usually the most severe, will go undetected possibly causing damage to the turbine.

3. Ever-changing environmental conditions may mask fault signatures, making problem detection difficult. Some sensors, like the vibration sensor, capture data which are representative of both the machine’s health *and* its operating environment. Sometimes, those parts of the sensor signal that are unrelated to the machine’s health (e.g. vibrations due to the rotational velocity of the turbine) complicate the interpretation of the sensor signal. It is therefore imperative that the techniques and algorithms employed within the MCM/PHM system are tolerant of the turbine’s environmental state and can be effective regardless of the operating conditions.

1.2 CONTRIBUTIONS

The contributions of this research relate to the development of a MCM/PHM system for ocean turbines. To the author’s knowledge, there were no prior publications related to the design and implementation of a software tool for ocean turbine monitoring. Research contributions are listed below.

- We address data fusion as a cross cutting concern of a condition monitoring system. To do so, we propose a data fusion approach to MCM/PHM systems designed according to the ISO-13374 standard.
- Some sensors, like the vibration sensor or accelerometer, emit thousands of readings per second possibly in the form of a continuous data stream. When

fusing these signals, one needs to consider the possibility that the data streams being combined may arrive at different times although they were generated at the same time. This research also proposes a formalization of barrier synchronization as a technique for coordinating sensor data streams prior to fusion.

- We empirically investigate machine learner performances and data fusion for reliable ocean turbine state detection from vibration data. In these studies, we also analyze the effect of missing data on classifier performance and consider the use of feature selection to reduce the training dataset size on feature level fused data.
- As previously mentioned, faulty or abnormal data are relatively rare compared to data characteristic of normal operation. Comprehensive experimental studies investigating the impact of class imbalance on machine learner behavior (with and without data fusion) in the condition monitoring context are also performed.
- Finally, we offer a Condition Monitoring Software System (CMSS) tool for ocean turbine MCM/PHM. In its current state, the CMSS tool performs data manipulation and state detection – two basic requirements of a condition based monitoring system per the ISO-13374.

1.3 DISSERTATION STRUCTURE

This dissertation is organized as follows. In Chapter 2, a background into the SNM-REC ocean turbine prototype and into MCM/PHM is provided. That chapter also describes a computer-controlled 20kW dynamometer designed for testing the components of that ocean turbine prototype. Related work regarding data mining, data stream management and data fusion in a reliability analysis context is surveyed in

Chapter 3. An overview of the data collection process, vibration analysis techniques, machine learning algorithms, data mining techniques and performance measures utilized in our experiments is given in Chapter 4.

The following chapter (Chapter 5) presents the proposed data and knowledge fusion framework as well as the barrier synchronization approach. Preliminary studies analyzing and comparing multiple data mining and data fusion techniques for enabling reliable state detection from sensor data are presented in Chapter 6 while more intricate experiments involving more sensors, missing values, feature selection and class imbalance follow in Chapters 7 and 8. In Chapter 9, we unveil the CMSS software tool for ocean turbine MCM/PHM and finally, in Chapter 10, conclusions and opportunities for future work are given.

Chapter 2

Background

A turbine is a rotary device with propeller-like blades which are driven by a continuous stream of a fluid (e.g. gas, wind, water) converting the kinetic energy of that stream into mechanical energy to produce electricity. Research and development of a 20-kilowatt ocean turbine prototype for harvesting ocean current energy from the Gulf Stream is underway by the Southeast National Marine Renewable Energy Center (SNMREC) at the Florida Atlantic University [4]. The goal of the center is to provide solutions for Florida's clean energy initiative by assessing the potential of converting ocean currents and ocean thermal energy into a reliable source of electricity while minimizing environmental impact.

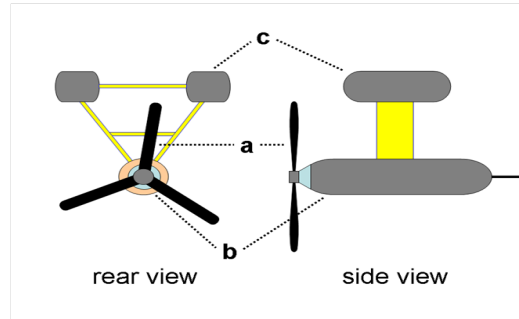


Figure 2.1: Closeup of nacelle

In this prototype, the turbine is housed within a pressurized enclosure called a nacelle (component *b* in Figure 2.1) which is connected to two pressure buoys (to con-

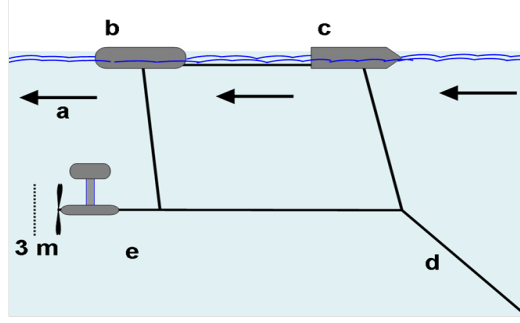


Figure 2.2: Turbine and its moorings

trol its pitch, yaw and roll) (2.1c) and a three blade propeller (2.1a). This structure, depicted as component (e) in Figure 2.2, is connected via cabling to component (b) – a monitoring and telemetry buoy (MTB) which is empty in the middle for buoyancy – and a barge for keeping the system upright (c). The underwater structure is tethered to the ocean floor (d). The northbound flow of the Gulf Stream is shown as (a) in Figure 2.2.



Figure 2.3: Picture of the Dynamometer

As a part of its research efforts, the center has developed a computer-controlled 20 kW dynamometer (Figure 2.3) for the purpose of testing the components of the ocean turbine [30]. A dynamometer is a device used for measuring force, torque or

power. They have been widely used in many industries, where applications include testing brake wear and friction for airplanes [27] and automobiles [2], measuring tool wear based on cutting force signals in machining systems [87], load testing for gas turbines [125] and wind turbine testing [129]. The structure of the dynamometer is discussed in greater depths in Section 2.1.

Careful and complete testing must be done to ensure the success of the ocean turbine project. System tests such as drive-train endurance testing, component testing, gear-box testing and fault simulation testing are made possible through the dynamometer. Additionally, data gathered from the dynamometer during testing can be used for designing and fine tuning an MCM/PHM system necessary for monitoring the turbine.

2.1 COMPONENTS OF THE DYNAMOMETER

The structure of the dynamometer under development at SNMREC is shown in Figure 2.4. In this diagram, we see four main components: a motor (MTRX), two gearboxes (GBXA and GBXB) and a generator (GENX).

The motor (MTRX) is a 3-phase induction motor which simulates the effects of ocean current on the machine. This component is powered by the electrical signal from the grid which is conditioned by two variable frequency input drives (not shown in Figure 2.4) via a multi-step process. First, a rectifier converts the AC signal to DC. A series of capacitors and bridges then feed the DC signal into an inverter to determine the frequency, voltage and current to the motor. The DC signal is then converted back to AC, which is then fed into the motor as the input signal.

The motor MTRX is connected to a gearbox GBXA which reduces the rotational speed of MTRX by a 21.8:1 reduction ratio. This reduction is necessary because

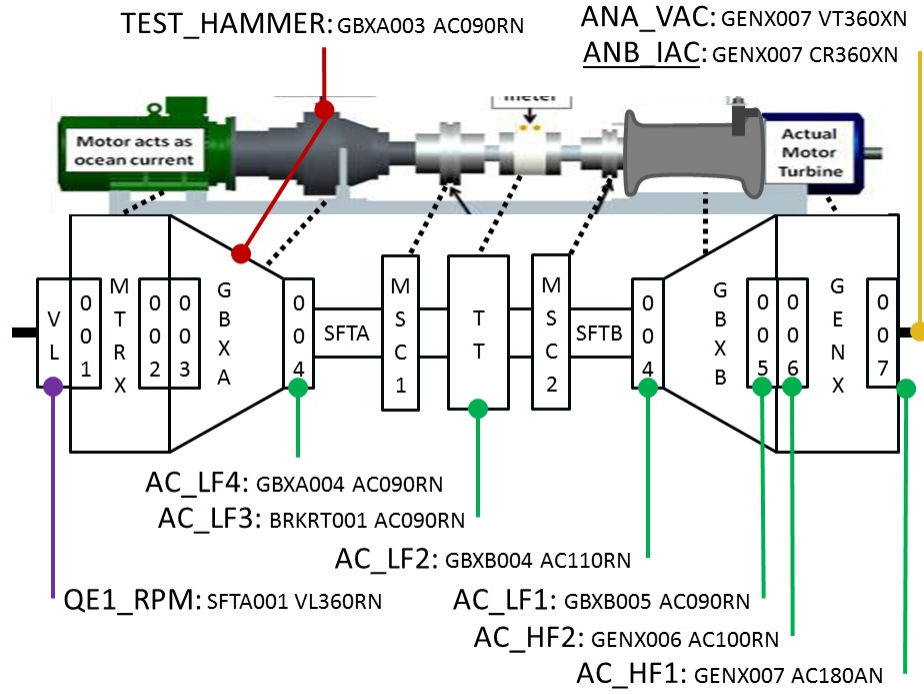


Figure 2.4: Dynamometer Diagram Showing Sensor Locations

the MTRX delivers torque and power at rotational speeds which greatly exceed that which can be handled by the shaft [148]. The dynamometer side drive shaft SFTA couples to turbine drive shaft SFTB via couplings MSC1 and MSC2. Both SFTA and SFTB rotate at this reduced rate. Planetary step-up gearbox GBXB has a 1:25 gear ratio and supplies an increased rotation speed to the AC induction motor GENX. By supplying sufficient torque to GENX, an electrical current is produced. ANA_VAC and ANB_IAC are phasor measurement units for measuring power quality.

Components labeled 001 through 004 to the left of driveshaft SFTA and 004 through 007 to the right of driveshaft SFTB in the same figure represent the bearing assemblies. These are numbered relative to the prime mover. Torque transducer TT records the torque of the driveshaft. Velocity transducer VL (i.e., encoder) measures

the rotational velocity of the input force to the MTRX. Motor shaft coupler MSC1 joins the dynamometer to a portion of the driveshaft that functions as a test sleeve for the torque transducer. MSC2 joins the test sleeve to drive shaft SFTB for the machine under test. Once power curves have been developed, the test sleeve will be replaced with the brake assembly.

Six single-axis accelerometers (2 high frequency AC104 sensors and four low frequency AC136-1A sensors) were mounted in different places on the dynamometer to allow for the acquisition of raw vibration data. Locations of these sensors are shown in Figure 2.4. For our experiment, we refer to these accelerometers as channels. The four low-frequency accelerometers, AC_LF4, AC_LF3, AC_LF2 and AC_LF1, are located closest to the prime mover and are channels 1 through 4 respectively. AC_HF1 and AC_HF2 are the high frequency vibration sensors which record the vibration from more rapid rotations by GENX, and are denoted as channel 6 and channel 5, respectively. All sensors except AC_HF1 were installed at or around 90 degrees relative to axial view of the unit circle with prime mover in the foreground (i.e., MIMOSA Convention). AC_HF1 was placed at 180 degrees. Other types of sensors, including leak sensors, oil quality sensors and thermometers, are not shown in the diagram and were not included in the experiments as configuration and testing of these sensors is still underway.

The forces exerted on the dynamometer will be similar in nature to those that would act on the turbine during an ocean deployment, thus sensor readings recorded from the dynamometer will be useful in developing an MCM/PHM system for monitoring the ocean turbine once deployed. The proposed MCM/PHM system will be data-driven, meaning that the collection and analysis of data and relevant background information (including historical information and expert knowledge) support the decisions made regarding the health of the turbine. The following section discusses

MCM/PHM in greater detail.

2.2 MCM/PHM

Machine Condition Monitoring / Prognostics Health Monitoring (MCM/PHM) systems are widely used in many domains to continuously assess the health and remaining useful life of complex machinery. Such systems record and interpret data from sensors attached to different components of the machine to detect faults, predict future failures and generate advisories for improving the lifetime of the machine.

To determine the condition of the machine, measurements such as the mechanical vibration levels, oil debris, temperature, pressure, angular velocity and flow around the ocean turbine need to be constantly recorded and analyzed. This is made possible by many types of sensors including submerged video cameras, pressure sensors, temperature sensors, leak sensors, an inertial mass unit (IMU), a tachometer, vibration sensors and electrical output sensors. In one sense, non-vibration data allows for quicker detection of new faults since vibration readings are typically sampled intermittently in bursts. In another sense, non-vibration data *trails* vibration data since vibration data reflects immediate physical forces while non-vibration data like oil temperature reflect convection processes that have occurred only after the rolling element had been damaged.

One way in which these sensor signals are used is for intelligent problem detection (*i.e.* automated fault localization, detection and classification). Fault detection is the process of identifying when a fault has occurred. Pinpointing the type and determining the location of the fault are the goals of fault classification and fault localization respectively. By providing information about the type and location of a fault, an operator can determine whether a maintenance expedition is necessary and exactly

which tools or parts are required to correct the problem. Also, detecting faults as soon as they occur allows for quick remediation (such as self adjustment or shutdown) which prevents damage to the turbine.

Proposed by the Machinery Information Management Open Standards Alliance (MIMOSA), the Open Systems Architecture for Condition Based Monitoring (OSA-CBM) specification ¹ [103] implements the ISO-13374 [74] standard which defines six key functional areas of a condition monitoring system along with the data structures and interfaces required for each. The six blocks or modules in the OSA-CBM model are described below.

2.2.1 Data Acquisition (DA)

The first step in the condition monitoring process – data acquisition – refers to the collection and digitizing of the data from sensors attached to the machine being monitored. Sensors such as thermometers, thermocouples, leak sensors and oil quality sensors produce a single reading periodically, while others like the vibration sensor (or accelerometer) output thousands of measurements per second. All data must be appropriately timestamped and calibrated. The output of the DA block, therefore, is timestamped, synchronized, digitized data.

2.2.2 Data Manipulation (DM)

In the data manipulation block, the digitized sensor output from the DA block are transformed into the desired format. Typically, signal processing (using techniques like Wavelet transforms [152], Fast Fourier Transforms [154], cepstrum analysis, filtering and windowing), time synchronous averaging [114], algorithmic computations

¹Available for download on the Machinery Information Management Open Systems Alliance (MIMOSA) website - <http://www.mimosa.org/>

and feature extraction are performed here.

2.2.3 State Detection (SD)

At this stage, the output from DM and DA are compared against the anticipated baseline profile values to determine the current state of the system as one of a predefined enumerated set of states (*e.g.* system normal, level high, alarm, alert, *etc.*).

2.2.4 Health Assessment (HA)

Health Assessment (HA) involves combining the outputs from the DA, DM and SD blocks with the output from other HA blocks to diagnose system faults and determine the health of the overall systems.

2.2.5 Prognostics Assessment (PA)

The life expectancy and future health of the system are projected, and the probability of future faults and failures is predicted in the Prognostics Assessment phase.

2.2.6 Advisory Generation (AG)

In the Advisory Generation block, reports on existing or predicted conditions along with advice on how to optimize the life of the machine are generated.

2.3 OVERALL ARCHITECTURE

Figure 2.5 graphically depicts the architecture and data flow through the MCM/PHM system for this ocean turbine prototype. In the diagram, yellow arrows represent the current links among the components of the system while gray arrows are tentative links. The ocean turbine itself is depicted in pink in the lower right corner of the

diagram. An onshore test platform for the components of the ocean turbine is also shown; the gray box labeled Ocean Current Emulator refers to this dynamometer testbed. Once development and testing of the turbine components is completed, the Ocean Current Emulator box will no longer be included as a part of the MCM/PHM system architecture.

There are three main sub-systems: the wet-side system (bottom of the diagram), the topside system (center) and the shore-side system (top of the figure). The wet-side system will be attached underwater to the turbine and is comprised of sensors, a Wavebook Data Acquisition (DAQ) unit and a safety controller. The sensors are mounted at different points along the turbines and are of different types. There are: five motor temperature sensors, a water temperature sensor, leak sensors, an RS232 oil quality sensor, multiple vibration sensors and a tachometer. The vibration sensors and tachometer are connected via Ethernet to the Wavebook DAQ which is responsible for the acquisition of vibration data. The remaining sensors are connected to a Wet Safety Controller, which is a part of a larger system, namely the Safety Control and Monitoring System (SCMS).

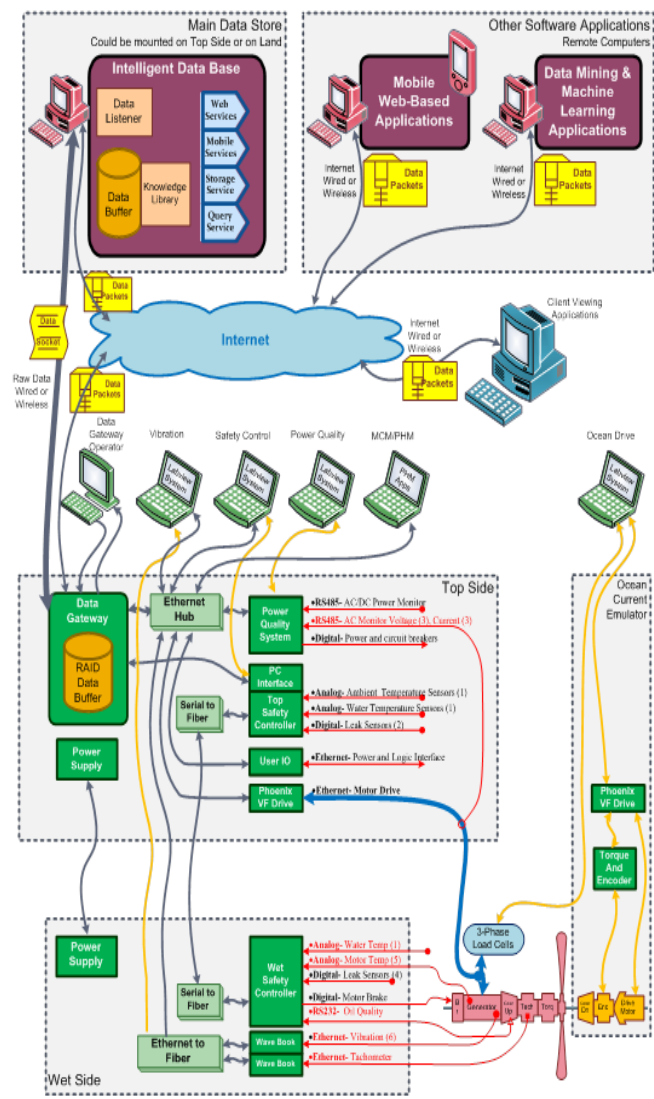
The SCMS is a dual data acquisition and control system which consists of two safety controllers – one of which resides wet-side (Wet Safety Controller) and the other located topside at the turbine’s power converter (Top Safety Controller). These controllers were designed so that they operate together with a single serial communications link but can act independently if there is a break in that link. The SCMS is responsible for performing state detection based on input from up to 24 sensors (12 sensors maximum per controller). It is compatible with thermocouples, thermistors, leak detectors, contact sensing, incremental encoders, load cells, IMU, pressure sensors, GFI, oil condition sensors, embedded circuits (RS485/422/232), voltage sensors and current sensors. If the sensor output exceeds safety limits (predetermined

by engineers and domain experts), the SCMS will trigger an alarm and can safely disconnect the power and shut down the machine.

Topside, a Data Gateway and wireless communication equipment for transmitting data back to shore will reside alongside the second SCMS controller. Cabling will allow the transfer of data between the topside and wetside subsystems. The Data Gateway sub-system captures data and instructions from the SCMS, Power Quality System and other such sources, providing a unified access interface. It will also be responsible for time-stamping and synchronizing sensor signals. A RAID (redundant array of independent disks) data buffer will be used within the Data Gateway providing redundancy to prevent data loss due to disk failure and enhancing performance. Voltage and current sensors (sampled at high speed) will interface with the Data Gateway via a voltage and current acquisition board (VIAB) within the Power Quality System. Additional sensors such as ambient and water temperature sensors and more leak sensors will also be located topside.

As the name implies, the shore-side components reside on land; they communicate with the topside system via an overseas wireless communication link. Further details regarding the communications infrastructure can be found in [12]. The Data Store (for storing raw and processed data along with maintenance, alarm and advisory history) and the MCM/PHM software can be situated either shore-side or topside. If topside, the Data Store and MCM/PHM software will be connected via the Ethernet hub. User interfaces allowing engineers to monitor feedback from the MCM/PHM system and to adjust operational parameters will be on land.

Figure 2.5: High Level MCM/PHM Architecture Showing Data Flow



Chapter 3

Related Work

Reliability analysis is the study of the ability of a component or system to operate under given conditions for a specific period of time according to some set of performance requirements. It involves determining the possible points of failure within a system and identifying those components which detract most from the system's reliability. This field encompasses multiple research domains including software reliability (the measure of the quality of a software design), reliability prediction (the forecast of the failure rate of a system) and failure analysis (the determination of the cause and consequence of a system failure). The failure rate or failure ratio, according to [73], is the proportion of the number of failures to a given unit of measure, such as failures per unit of time, failures per number of transactions, or failures per number of revolutions. The use of the term "reliability analysis" will henceforth pertain only to failure analysis and reliability prediction.

Automated reliability analysis has been made possible by the availability of cost effective sensor technology and increased computing performance. Continuing reductions in cost and increases in functionality of sensors offer unprecedented opportunities to measure multiple aspects of a system to assess its reliability based on a wealth of data. Machine Condition Monitoring / Prognostics Health Monitoring (MCM/PHM) systems utilize sensors attached to the machine being monitored to enable real time

health assessment, prognostics and advisory generation. To process and analyze this data, a combination of machine learning and data fusion are needed. The concept behind data mining and machine learning is defined in Section 3.1 where a literature review of issues pertaining to handling sensor data streams (which are produced by some types of sensors) is also included. A discussion on data fusion follows in Section 3.2.

3.1 DATA MINING & MACHINE LEARNING

Machine learning is a field of study focused on the design and development of algorithms, called classifiers or machine learners, that allow computers to predict a value for or classify/label new instances (or observations) based on patterns discovered in data supplied to those algorithms. The training set is the set of observations that are used to initially train or build the classification models. In some cases, the actual class to which each observation belongs is already known. The test set consists of new observations that the models are applied to. Each observation is then labeled by the model as being representative of one or more states (classes) from the set of possible classes C . The machine learning algorithms and data mining techniques employed in the various experiments will be discussed further in Chapter 4.

In the next section, however, we consider a branch of data mining dedicated to inferring knowledge from a continuous stream of data. Data stream mining, as this sub-branch is called, is also an integral part of MCM/PHM as some sensors, such as vibration and current sensors, produce a continuous feed of measurements, called a data stream, instead of a single periodic reading. These data streams can be analyzed to determine the state of the component being measured and the presence of faults.

3.1.1 Data Stream Mining

Over the years, data stream mining [38] has been the focus of many research papers and is used in many domains including fraud detection [110], medical monitoring [78] and stock market analysis [139]. This section examines the challenges associated with mining and storing data streams and surveys research efforts which address some of these issues. By exploring these issues and possible solutions, we hope to highlight additional research opportunities in this area.

Data stream mining techniques can be applied in several ways in reliability analysis. One way in which data stream mining is used in reliability analysis is in the mining of *changes* in the sensor data. Mining changes in sensor data may provide useful information regarding the state of that sensor and reduce conflicts in knowledge fusion systems [40]. Knowledge fusion systems attempt to integrate data from multiple sensors to deduce the state of a component in a system or of the entire system itself. Another important aspect of reliability analysis for which data stream mining has proved itself useful is survivability. The survivability of a system is its ability to operate efficiently in the presence of catastrophic failures [100]. Survival analysis is used to represent the time to an event, such as the failure time of a mechanical system.

Data stream mining encompasses techniques which utilize information gathered from past cases to label incoming data for prediction or classification of faults, and those which identify faults based on their similarity to a previous case using clustering. In the following section, we review issues plaguing data stream mining and management as it relates to reliability analysis.

Concept drifts and model updates

Knowing when to update a mining model is one of the challenges in data stream mining. A mining model could be updated periodically, incrementally (with every change in data) or reactively (rebuild the model only when it no longer suits the data) [161]. Frequent updates of the mining model wastes resources on insignificant changes while infrequent updates risk model inaccuracy and the resulting system degradation [161].

To complicate this issue, the underlying concepts in a data streams have a tendency to change over time. Mining concept-drifting data streams has been the focus of many articles, including [53, 68, 88, 158]. Algorithms adapted to concept drifting streams include incremental decision trees such as the concept-adapting very fast decision tree [68] and ensemble classifiers such as the streaming ensemble algorithm (SEA) [68].

Adjusting the mining model to adapt to changes in the state of the system being monitored must also be considered. Inaccessible machines, for example, may have to operate with non-critical faults until a scheduled maintenance date when the system state would revert to normal. If the classifier was trained based on data generated while the system was in an abnormal state, a recurring fault introduced after the system has been returned to normal may not be correctly classified by the mining model since the classifier may now consider the existence of that fault as a part of the normal operation of the system.

Selecting an appropriate mining method

In general, data stream mining approaches generate a model from historical records and use this model to classify new instances in the data stream. Such a method

permits only a single pass through the training data since a second scan of the data is infeasible due to the high rate of incoming data, and will need to incorporate new data from the stream as appropriate. Some data mining techniques such as decision trees do not adapt well to handle continuous data [5], but others such as ensemble classifiers [92, 156], one-versus-all classifiers [68], and k-means clustering [113] have been applied successfully to data stream mining.

The possibility of redundancies within ensemble classifiers and the inability of single components of incremental classifiers to be updated independently when a concept has changed are challenges which must be addressed. Redundancies within ensemble classifiers can be avoided using pruning techniques such as instance based pruning [156] to identify the subset of classifiers that produce the same results as the entire ensemble. The challenge of determining which classifier within an ensemble is to be updated is discussed in [68] but remains an open issue.

Model over-fitting

Model over-fitting, which occurs when a mining model is too specific or is too sensitive to the training dataset that was used to generate that model, is more likely to occur in streaming environments. This may be because of a lack of training data and possible biases in the training dataset resulting from the data originating from a single source [158]. Traditional data mining techniques such as cross validation are not well suited for data streams because they require more than a single pass over the training data. The framework proposed by Wang *et al.* [158] addresses the over-fitting problem by harnessing concept drifting patterns. Efficient feature selection algorithms such as [80] will also reduce the risk of over-fitting.

Cost sensitive learning

In reliability analysis, the cost of a false positive (or false alarm) is typically not equal to the cost of a false negative. Inaccessible systems, for example, have a low tolerance to false alarms because of the expenses associated with retrieving the equipment for repair. One approach to cost sensitive learning from data streams involved weighting classifiers within an ensemble based on their mean square error [156].

Imbalanced datasets in classification problems

In reliability analysis, it is typical for the dataset to be imbalanced, meaning that the ratio of positive (faulty) instances to negative (no fault) instances is skewed in favor of the negative instances. Learning algorithms such as decision trees have been known to perform poorly on imbalanced dataset problems because of their tendency to classify all instances as negative to maximize accuracy. Hashemi *et al.* [68] investigated the use of a novel under-sampling scheme for their ensemble classifier which restricts each negative instance in the training set from being used in the models of more than two classifiers within the ensemble. This scheme, along with techniques such as [146] and [140], could address this issue.

Missing or incomplete data

Missing or incomplete data is not uncommon in sensor networks, so any stream mining algorithm or framework should make provisions for handling missing, delayed or out-of-sequence data. Budhaditya and colleagues [8] proposed a framework based on a compressed sensing (CS) theory [29] for detecting anomalies in incomplete data by either sampling a subset of the sensors or of the number of frames in a temporal stream. Three techniques for substituting values for missing data points, namely

Bayesian multiple imputation, k -Nearest Neighbour imputation and Mean imputation were investigated by [85] as solutions to the missing data problem.

Modeling changes in mining results

Observing temporal changes in data streams may provide useful information about the system. By sensing the fluctuation in the data stream from a particular sensor, for example, it may be possible to identify that a sensor is failing based on increasing variances as time progresses. Algorithms such as *MAIDS* [11] have been designed for this purpose.

Data preprocessing

The design of a lightweight preprocessing technique which can guarantee the quality of the mining results remains as an open problem in data stream mining [58].

Formalizing data stream computing

Formalizing data stream mining within a theory of stream computation enables the design and development of mathematically sound stream mining algorithms [57]. A formalization of data streams in signal processing traditionally [56] uses Z-transforms – a discrete version of Laplace transforms. An approach geared to the composition, or the gluing together, of software components was more recently proposed in [127]. Using coinductive stream calculus to define signal flow graphs, this approach may make integration of learners into a stream processing environment easier to implement.

3.1.2 Data Stream Management

Centralized vs. decentralized processing

Processing live data may be either centralized or distributed (i.e. permitting some computation to take place at the individual sensor site). With centralized processing, the amount of bandwidth required to transmit all the data continuously (and in real time, in most cases) to a central storage system for immediate querying must be considered. With a distributed approach, each sensor node has the capability of processing its own data and then transmitting the results to a centralized location for further analysis. In distributed sensor data management, it is important to consider the possibility of data loss during transmission as well as how processing and storing its own data will affect the power consumption at a node. These concerns can create a bottleneck for the performance of the entire system.

Data aging

Another challenge to data stream management is the timely identification of stale data. An optimal data aging strategy would need to determine which data in the training set is no longer relevant, such as those pertaining to previous concepts in a concept drifting stream. As explained in [156], discarding data after a predetermined length of time may lead to some interesting problems. If the selected lifespan L is large, then the possibility of having outdated instances within the training dataset is high; if L is small, the risk of overfitting is greater since there may be insufficient instances in the training dataset. The aging strategy presented in [156] provides a workaround by considering the class distribution in addition to the arrival time when selecting the data lifespan.

Storage space restrictions

The continuous flow of a data stream demands an unlimited storage system to hold unprocessed streaming data and/or the results of the mining operation. In most cases, there is minimal storage at sensor nodes, so data storage may be centralized. Novel stream management systems (e.g. StonesDB [25], DIMENSIONS [59], TinyDB, Cougar and Diffusion) were designed to provide efficient storage solutions for data streams.

Limited availability of resources

One of the biggest stream management issues relates to the lack of available resources including storage space, processing power, network connectivity and energy. In some domains, including ocean systems, available network connectivity may be limited, spotty and incapable of sustaining high data transfer rates. More reliable network connections tend to be difficult to acquire or expensive, and may not be feasible offshore in the case of oceanic systems. An ideal solution would therefore need to minimize communication overhead.

Because of the constant transmission of data, the lifetime of the battery in a battery-operated sensor is greatly reduced. A novel approach to resolving this issue in an ocean turbine is to wire the sensors in to the turbine's battery pack which can be recharged by the flow of ocean currents. Other solutions attempt to conserve energy consumption by reducing the quantity of data to be transferred or by transmitting the data in batches. By reducing the volume of data, the amount of time the sensor spends transmitting data is also decreased. The volume of a data stream can be reduced via well known summarization methods: sampling, load shedding, aggregation and approximation.

Sampling techniques in general involve altering the number of data points to achieve desired results, and are typically used in class imbalance problems to alter the class distributions. Under-sampling techniques, which involve preserving only a subset of the negative (not faulty) instances from the original data stream, reduces the size of the data stream by ignoring or discarding some of the original data points. In load shedding, a sequence of items within the data stream are discarded. While it has been applied successfully to data streams, it presents the same problems as sampling [58]. Aggregation employs statistical measures such as average, maximum, minimum, etc. while approximation techniques [23, 24, 112] involve replacing the original data stream with an approximated signal (with error bounds) which is tailored for the application domain [24].

Handling the continuous flow of data streams

Passive systems such as traditional database systems typically have high latencies due to the cost of a database storage operation and constant polling for data [139]. Active stream processing engines (SPE) which utilize specialized primitives and constructs like time sliding windows to perform on-the-fly processing of streaming data but store data only when necessary are popular solutions to this problem. The querying language used within an SPE typically resembles SQL for databases, and often includes constructs permitting joins and cross querying of multiple streams. StreamSQL is one of the most popular variants of SQL and was specifically designed to express queries on continuous data streams.

Another issue pertaining to the continuous flow of data streams is the inconsistency in data transfer rates. Algorithm output granularity (AOG) is a resource-aware data analysis approach capable of handling very high data rates which performs data analysis locally on a resource constrained device. AOG works by performing mining

operations, followed by adapting to the data stream rates and resource availability, and then by merging knowledge structures when the memory is filled past a certain capacity [58].

Merging distributed streams

Typical monitoring systems consist of sensors installed in different locations on the machine or system being monitored. The resulting data streams must be merged, sometimes in real-time, to allow an overall system analysis. One approach to merging distributed streams was based on a common key [102]. Associated with merging of data streams is the topic of data fusion which is surveyed in the next section.

3.2 DATA FUSION

Finding an appropriate and effective means of aggregating data and/or information from multiple sources is a challenge arising in many applications today. The term *data fusion* may have been coined as early as the 1970s to represent the process of combining data from multiple sources for the sake of building a more complete and accurate picture, while typically reducing the quantity of data or information that is returned as output. The usage of the term data fusion here encompasses multi-sensor data fusion, data integration and knowledge fusion.

As the name implies, multi-sensor data fusion (MSDF) is the fusion of multiple sensor signals, whether physical (real) or virtual. A physical sensor directly records and transmits measurements of some physical process. By contrast, a virtual sensor produces a signal by using mathematical models to estimate readings for a particular property or process condition based ultimately on data from physical sensors.

The term data integration as it is used in this study refers to the process of merging

data or information from multiple sources to form a complete view. The challenge here involves combining data from multiple sources which may have differing conceptual, contextual and typographical representations so that all the data in the input is somehow reflected in the output in a unified format.

Knowledge or information fusion is the combination of predetermined facts, states or situations to produce a complete, more accurate interpretation of the available information. Fusing higher order information poses unique problems, including varying source credibilities and conflicting information. This type of fusion is often used to support human decision about the system or to enable automated decision making.

Systematic implementation of the entire data fusion process typically follows a particular design or architecture – such as those discussed in [39, 50, 99] – and incorporates mathematical algorithms (implemented in “fusion engines”) for combining the data. We will review some of these algorithms in the next section. A well-designed data fusion system should consider the likelihood of imperfections in the data such as uncertainty, imprecision, vagueness and incompleteness while considering possible inconsistencies and data correlation among sources.

Consider a set of p possible outcomes or classes $C = \{c_1, c_2, \dots, c_p\}$ and a set S of n sources where $s_i(t)$ is the output from source i at time t . For discrete valued $s_i(t)$, commonly encountered in knowledge fusion problems, $s_i(t) \subseteq C$, meaning that the output of the source is some single class or subset of classes. For continuously valued $s_i(t)$ such as in MSDF of sensors that produce numeric readings, $s_i(t) = V_i = \{v_1, v_2, \dots, v_q\}$ such that $\forall v \in V_i, v$ is a real number and q depends on the type of source or sensor. If source s_i is a global positioning system sensor (GPS), for example, $q = 3$ and the vector V_i would represent the location triple – latitude, longitude, and altitude. Given the above convention, we can define the four previously listed characteristics of data as follows:

3.2.1 Uncertainty

Uncertainty in a source's output is that source's inability to state without a doubt that its statement or output is accurate. If ϑ_{s_i} represents the certainty of source s_i , that source is said to be uncertain if $\vartheta_{s_i} < 1$. This can be represented in the discrete case as: $\exists c \in s_i(t) \subseteq C$ where there is some doubt that c is true; for the continuous case, we get: $\exists v$ in $s_i(t)$ where there is some doubt that v is an accurate reading.

Uncertainty may differ for each observer or source. So, it is possible that for any two sources s_i and s_j in S , the uncertainty in the set of statements or data output by s_i at time t is different from the uncertainty that s_j has in its observations at that same point in time, meaning that the uncertainty in s_i is not necessarily equal to that of s_j .

3.2.2 Imprecision

Imprecision occurs when the information contained within a statement or source could have multiple possible values for one physical phenomenon. It can therefore be defined as either a lack of exactness or of a proper definition.

The imprecision of a source whose output is a continuous value can be defined as a single value ϵ , such that the output of that source is expected to fall within $\pm\epsilon$ of the true value. In mathematical terms, the imprecision of a source's reading $s_i(t) = \{v_1, v_2, \dots, v_q\}$ at time t is represented by: $\{v_1 \pm \epsilon, v_2 \pm \epsilon, \dots, v_q \pm \epsilon\}$.

In a discrete case, this is represented as $s_i(t) \subseteq C$ where $|s_i(t)|$ exceeds the expected number of outcomes or statements that a source should produce. The degree of imprecision is therefore representative of how much $|s_i(t)|$ exceeds the expected number of outcomes or statements. For example, say we have a standard die with 6 faces; if rolled, the possible values of the die are $C = \{1, 2, 3, 4, 5, 6\}$. Assuming that

the die is not allowed to fall on its side (such that multiple values are exposed), an observer of the die being rolled at time t is expected to state the single face value of the die after it was rolled. An example of an imprecise statement that can be made by an observer s_i about that particular roll of the die would be that the die stopped on either a 4 or a 5, or in mathematical notation, $s_i(t) = \{4, 5\}$. Imprecision occurred here because the observer was unable to distinguish between the two values 4 and 5.

3.2.3 Vagueness

Vagueness is a specific type of imprecision where a decision or statement from a source only has a partial degree of truth. Mathematically speaking, vagueness occurs whenever there is a one-to-many relationship between the observations made by the source $s_i(t)$ and the set of all possible outcomes C . This is only apparent in discrete cases. For example, if a source s_i is to make a determination of the state of a glass of water (i.e., is the glass full or empty or $C = \{full, empty\}$) where the capacity of the glass is 16 millilitres and the volume of water in the glass is 4 millilitres, this source may output that the glass is 1/4 full (or alternatively, the glass is 3/4 empty). So, $s_i(t) = \{full, empty\}$.

3.2.4 Incompleteness

Incompleteness occurs when a source cannot provide all the required information. For example, a video camera may provide incomplete data about a room where a part or all of its view is obscured. For continuous cases, incompleteness in data from source s_i at time t is given by $s_i(t) = V_i = \{v_1, v_2, \dots, v_q\}, \exists v \in V_i$, where v is undefined, i.e. $v = ?$.

For discrete cases, incompleteness can be represented as $s_i(t) \subseteq C$ where $|s_i(t)|$ is less than the expected number of outcomes or statements that a source should

produce. Incompleteness also occurs if $\exists c \in s_i(t) \subseteq C$ where c is undefined, i.e. $c = ?$.

3.2.5 Inconsistencies

Inconsistencies in data arise when multiple sources present conflicting or discordant information on the same phenomenon. This is especially a problem in knowledge fusion where information produced by multiple, possible disagreeing sources must be combined.

We can define the inconsistency between sources s_i and s_j at time t as follows. If $s_i(t) = c_f$, $s_j = c_g$, and $c_f \neq c_g$, then sources s_i and s_j are said to be inconsistent. Similarly, for continuous cases, given that $s_i(t) = V_i = \{v_{i1}, v_{i2}, \dots, v_{iq}\}$ and $s_j(t) = V_j = \{v_{j1}, v_{j2}, \dots, v_{jq}\}$, there exists one or more data points in V_i having a value differing from the corresponding data point in V_j by some threshold ϵ , i.e. $\exists k < q, v_{ik} \in V_i$ where $v_{ik} > v_{jk} + \epsilon$.

The degree of inconsistency between sources can be represented as $(1 - h)$, where h is the level of concordance between the sources. Inconsistency $(1 - h)$ can be reduced or removed by increasing the amount of imprecision [79].

3.2.6 Correlation

Sources s_i and s_j are said to be correlated if some relationship exists between their outputs. They are fully correlated if all the observations made by source s_i are represented in some form in the output from s_j ; in such case, data fusion is not needed [18]. A problem in many applications [111, 149], data correlation can occur especially in applications having multiple sensors measuring the same aspect of a system. Correlation can also occur within a source, such as among data points of its output data vector or among features in the output feature vector. When there is no correlation among sources, data points or features, these are said to be independent.

Researchers aiming to address the challenges associated with data fusion have developed and utilized numerous fusion frameworks and techniques over the years. Their work spans many domains including medical diagnosis [162], military defense and tracking systems [136], navigation systems for autonomous vehicles [22], remote sensing [13] and intrusion detection for cyber security [63].

This literary review focuses on data fusion in the context of reliability analysis. More specifically, we present approaches to data fusion which have been or could be adapted to reliability analysis. Several surveys of data fusion techniques have already been done, but these were specific to other domains such as image fusion [28] and target tracking [136] or to a particular application [10, 98].

3.3 DATA FUSION TECHNIQUES

Data fusion algorithms or techniques are often grouped by the level at which they are applied:

1. Data level fusion techniques [128] are applied on the raw data. Sometimes called pixel-level or signal-level fusion, this level of fusion involves combining raw sensor signals prior to performing any data transformations, feature extraction or data manipulation. In order to combine sensor signals at the data level, they must have originated from sources which produce the same type of signal.
2. Feature level fusion [64, 128] is performed after features or attributes, which are individual measurable characteristics of the data, are extracted from the raw data. After extracting the feature set, a set of features from each signal are combined to produce a fused signal. Unlike data level fusion, feature level fusion can be applied to data from both homogeneous and heterogeneous sensor types.

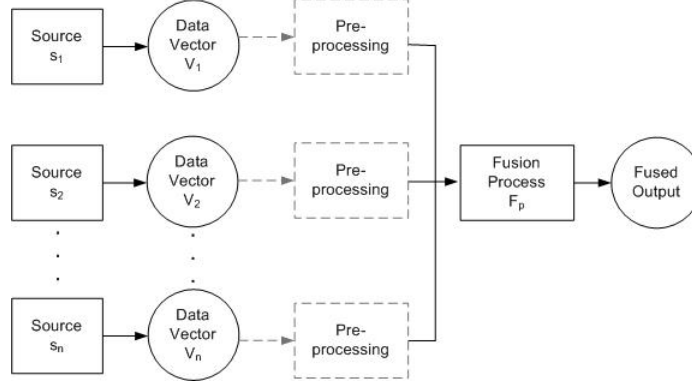


Figure 3.1: Data Level Fusion

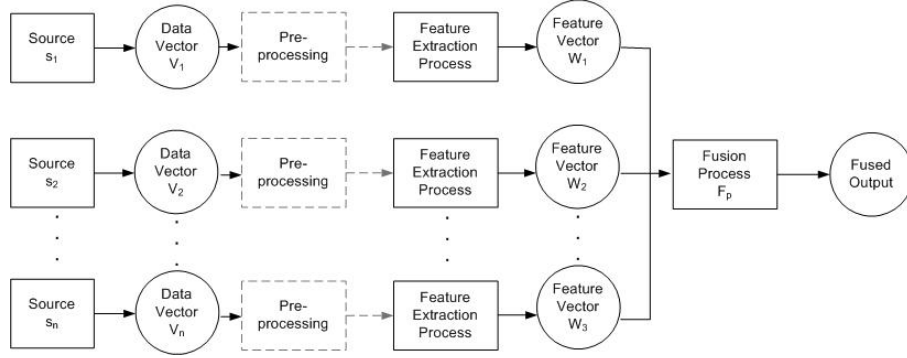


Figure 3.2: Feature Level Fusion

3. Decision level fusion [17, 64, 128] occurs while or after a local decision has been made from each source. It involves making a local decision from each signal and then combining the decisions to get the final output, usually with the aid of a mathematical model. This can be done by combining the decisions themselves or by combining the probabilities of class membership for each class and selecting the class with the highest probability [149].

Data fusion techniques can also be categorized based on the underlying theory or approach that is used to combine the data. In the following sections, we will discuss five approaches, some data fusion techniques which follow each approach, and how these techniques can be or have been applied to reliability engineering and analysis.

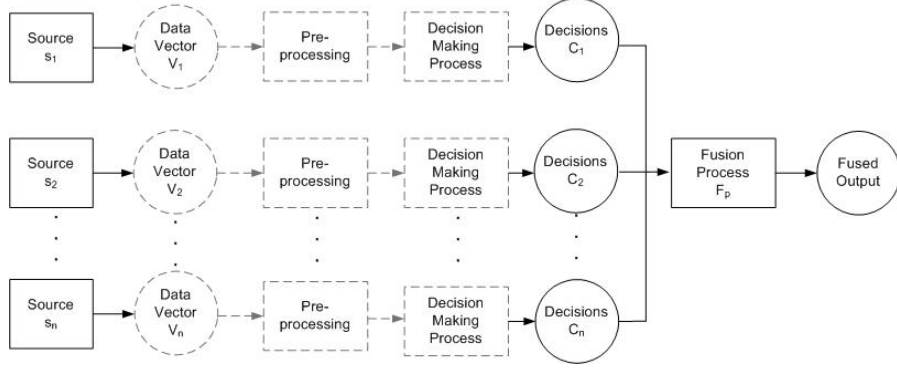


Figure 3.3: Decision Level Fusion

3.3.1 Bayesian Theory

In probability theory, each statement or event is assigned a probability. In data fusion problems, techniques based on probability theory are typically associated with Bayes' rule [51] – a special application of conditional probability where belief is quantified in a specific way. Probabilistic models are often used to model uncertainty in data. The Bayesian-based techniques discussed in this paper are a Bayesian fusion rule, hierarchical Bayes and Bayesian networks.

A class-specific Bayesian approach to fusion was applied to combining machine learners for fault diagnosis in automotive systems [20]. This technique involved summing the posterior probability produced by each machine learning algorithm or classifier for each class to determine the overall posterior probability of that class. This approach is based on the idea that different machine learners or classifiers may be better at identifying a particular class or fault than others.

A recursive substitution rule based on Bayes' rule, known as hierarchical Bayes, was applied to reliability analysis in [116]. In that study, hierarchical Bayes was used to model the probability of failure within a specified time span (failure intensity) and the effectiveness of repair for multiple manufacturing (assembly) lines.

A Bayesian network is a graphical representation of the probabilistic model where the Bayesian random variables are encoded as nodes and the conditional dependencies between these nodes as edges. They have been applied to data fusion for risk analysis [108], reliability analysis of military vehicles [107], reliability analysis of software in consumer electric products [109] and for modeling and analysis of the reliability of dynamic systems [6]. A commercially available tool for modeling these networks is the Bayesian Network software tool AgenaRisk by Agena Limited [1].

3.3.2 Possibility Theory

In some fusion problems, the a priori information that is required by the probability model is unavailable. The possibility theory model [31] is one alternative that does not require prior knowledge. Like probability theory, possibility theory is based on set functions and can model incomplete data. It differs from probability theory in its ability to model imprecision.

Four modes of combining data from multiple sources according to the possibility theory are typically used:

- **Conjunctive:** This mode of combination can only be used if all sources are considered to be equally reliable and non-conflicting. In this case, the source that is considered to be most informed about an outcome or event is the one which assigns the smallest possibility degree to that value [31].
- **Disjunctive:** This mode applies when the sources are not equally reliable, or are highly conflicting, and allows coverage of the entire domain provided by the individual inputs. This mode yields a less specific and logically weaker result than the conjunctive mode [115].
- **Adaptive:** To overcome the challenges associated with choosing an appropriate

mode of combination, adaptive modes of combination were created. The Dubois adaptive combination rule [31] is a progressive mode of combination whose behavior shifts gradually from conjunctive to disjunctive as the degree of conflict among sources increases or from disjunctive to conjunctive as the degree of conflict among sources decreases.

- Progressive: The progressive rule of combination [115] is a modification to the adaptive combination rule which gradually takes conflict into account.

Mourelatos and Zhou [106] apply a possibility-based approach for assessing reliability in system design and for performing reliability-based design optimization, which is a technique used for optimizing the design of an engineering system while accounting for reliability constraints.

3.3.3 Fuzzy Set Theory

Fuzzy sets are sets whose elements can have different grades or levels of membership to a class [96]. Features, sensors or classifiers are examples of elements. In contrast to a classical bivalent set or *crisp* set where an element will either belong to the set or not, fuzzy sets define a membership function which allows for a degree of membership to be assigned to each element, thus allowing for the modeling of incomplete or imprecise data.

Fuzzy set models have been widely used for data fusion in many domains over the years [96]. An instance of its usage for reliability analysis is given in [96], where a model based on a Choquet fuzzy integral and fuzzy measures was applied at both the feature level and decision level for fault diagnosis in rotating machinery.

Fuzzy *c*-means is a method of clustering where instead of belonging to a single cluster, each data point has a degree of membership to multiple clusters. It is similar

to the k -means algorithm [92], and shares the problem of the result being dependent on the initial choice of weights.

A study by [76] discussed the use of the fuzzy c-means clustering algorithm for fusing data from multiple sensors measuring the same phenomenon so as to separate faulty readings from normal ones. This eliminates inherent inaccuracies due to operational problems from sensor measurements.

Matlab[®] provides support for fuzzy sets through its Fuzzy Logic Toolbox.

3.3.4 Evidence Theory

Evidence reasoning, evidence theory, belief theory or Dempster-Shafer theory is a generalization of Bayes' rule (see Section 3.3.1) which can model impreciseness and uncertainty in data. Subjective probabilities – a probability that is derived from a source's determination of the likelihood of an event or outcome – are used to deduce a degree of belief in the source and a belief that an event or outcome occurred based on the reliability of that source [131].

Fabre, Appriou and Briottet [51] discussed using evidence theory for modeling sensor reliability in operational contexts that affect data acquisition, such as meteorological conditions, noise and surface variations. They then applied the Dempster-Shafer Rule of Combination (DS-ROC) [131] to combine this contextual information from multiple sensors during the decision making process.

A hybrid form of the DS-ROC was used by [116] for fault diagnosis. In this hybrid approach, the researchers use fuzzy theory to represent evidence sufficiency, which, based on their results, improved the performance of the DS-ROC algorithm for their application.

Transferable Belief Model. Suppose a machine condition monitoring (MCM) system employs two techniques, A and B , to identify faults as they occur and to

classify the type of fault as being either x , y or z . According to approach A , at time t , there is a 90% certainty that fault x occurred and a 10% chance that y occurred. For the same time t and given the same data, approach B produces a 95% probability that the fault z occurred and a 5% chance that y occurred. Intuitively, one would say that it is possible that none of the faults were present, but have a high degree of certainty that fault y did not occur. And yet, based on the DS-ROC and Bayesian theory, fault y certainly occurred. This counterintuitive result stems from the fact that y was the only fault type in common between techniques A and B .

Motivated by such pathological cases, the Transferable Belief Model (TBM) was developed by Smets in [134]. The TBM extends the evidence theory to allow for scenarios such as the one presented where, due to non-random uncertainty (caused for example, by partially reliable sources), an event not described within the power set of all classes, outcomes or events is possible.

In later work, Smets posed TBM as a data fusion approach and discussed three modes of combination – conjunctive, disjunctive and cautious conjunctive – of the degrees of belief assigned to some data or statement within the TBM [135]. The ideas behind the conjunctive and disjunctive modes remain the same as previously discussed. The cautious conjunctive mode, on the other hand, is used when the same thought or piece of evidence is conveyed by multiple sources, i.e., the sources are correlated. This approach was applied in [49] for determining sensor reliability.

Matlab[®] functions for the TBM are currently available on-line for download at <http://iridia.ulb.ac.be/~psmets/>.

3.3.5 Other

Neural Networks

An artificial neural network, or ANN, is a machine learning technique inspired by the way the human nervous system processes information. It consists of a number of interconnected neurons or nodes, each of which produces an output according to some function of its inputs.

An instance of the use of neural networks for reliability analysis is given by Fan *et al.*, who implemented Radial Basis Function Neural Networks (RBF-NNs) for MSDF of sun sensors, infrared earth sensors and gyros within a Satellite Attitude Measurement System (SAMS) to detect hardware faults in the sensors themselves and improve the measurements output by the SAMS [52]. RBF-NNs are included in the WEKA Data Mining Software Tool [67] and in the Neural Network toolbox for MATLAB®.

Kalman Filter

Typically used for data level fusion, the Kalman Filter (KF) is a mathematical model which uses the physical laws of motion, sensor or other type of measurements and known control inputs to a system to recursively estimate its state (such as a position) while minimizing the mean square error. A popular technique, it is included in MATLAB® release 7.0.1. KF assumes *a priori* knowledge of the process and measurement noise covariances and cannot be used when there is no specific statistical model of uncertainty [136]. A bank of Kalman Filters were used in [86] for fault detection and isolation for an aircraft gas turbine engine.

Voting & Summing

Voting and summing are two simple approaches to data fusion usually involving multiple machine learners.

In a voting approach, sources vote on the expected outcome and the outcome with the highest number of votes is selected as the output. A voting scheme was used in [120] for aggregating the outputs from four independent object tracking modules to create a more reliable approach for tracking a moving object.

Summing approaches involve applying a mathematical sum or average of weights, probabilities, etc. Summing typically outperforms voting if the error distribution is Gaussian but under-performs voting if the estimation error has a heavy tailed distribution [136].

Classifier Combination

As previously mentioned, decision-level fusion can be applied to merging information from multiple experts. When these experts are machine learning algorithms, such as in [20], this approach is called classifier combination (CC). This technique of combining machine learners to harness their individual strengths and reduce errors in classification or prediction is the foundation behind ensemble classifiers [81], and is sometimes referred to as stack generalization. The way this is done depends on the richness or *level* of information produced by the machine learner or classifier. These information levels are:

- **Abstract:** Learner outputs a single element out of the set of possible outcomes C . At this level, a *Behavior-Knowledge Space* method [126] or a voting technique (see Section 3.3.5) can be applied.

- **Rank:** Learner outputs an ordered list of elements with the topmost element being first choice. Class set reduction methods such as Intersection of Neighborhoods which aim to eliminate those elements that are least likely to occur and class set reordering methods such as the Borda Count method and Logistic Regression which combine the rankings from the different learners to produce an overall ranking can be applied at this level [126].
- **Measurement:** Learner assigns a numeric value to each element of C which depicts a degree of certainty or the degree that the sample has the label. Bayesian fusion was applied in [20] for the fusion of classifier output within the fault diagnosis system of an automobile. Parikh, Pont and Jones [117] applied Dempster-Shafer Rule of Combination to CC of three classifiers, namely Multi-Layer Perceptron, Radial Basis Function Neural Network and k -nearest neighbor, for condition monitoring and fault diagnosis of a diesel engine cooling system.

3.4 CONCLUSION

In this chapter, we reviewed relevant research pertaining to data mining and data fusion. This discussion was meant to provide any reader with an understanding of the fundamentals of these two crucial concepts and how they can be applied to reliability analysis in various contexts. For further reading as to how these techniques can be applied in other reliability analysis applications, refer to one of our published works [32]. A more extensive discussion of data fusion and applicable techniques was presented in another survey paper [41]. In the next chapter, we describe how data fusion and data mining are applied in our studies. More specifically, we give an overview of the data acquisition, manipulation, mining and fusion techniques employed while experimenting to find the optimal approach to the construction of a reliable ocean

turbine MCM/PHM system.

Chapter 4

Methodology

This chapter describes the methodologies utilized in the experiments conducted as a part of this research. More specifically, we discuss the data collection experiments, vibration analysis (data manipulation) algorithms, classifiers, data fusion approaches, data mining techniques including data imputation algorithms and feature rankers as well as the performance measures.

Experiments conducted during this research revolve around vibration data. Vibration readings of rotating machinery contain distinct signatures which can be used to determine the state of a machine. Irregular vibration signals within a mechanical system could be indicative of imbalance, looseness and distortion, misalignment, defective bearings, bad drive belts, misalignment, gearing and coupling inaccuracies, critical speeds, various form of resonance, reciprocating forces, hydrodynamic forces, oil whirl, friction whirl, rotor/stator misalignments, bent rotor shafts, defective rotor bars, *etc* [77].

4.1 DATA ACQUISITION

Data used in these experiments were collected from two types of rotating machinery: a typical 120 volts AC 50 cm box fan and the ocean turbine dynamometer described in Chapter 2. In both cases, vibration sensors mounted on the machine are used to

measure its vibration a certain number of times per time interval, called the sampling rate. These sensors are connected to a WaveBook Data Acquisition Unit which records the data and performs time synchronous averaging (TSA) [90] – a process which segments the data into equal length blocks related to the different rotational phases and averages the blocks to reduce noise – of the vibration signals. In TSA, interpolation is used to introduce new data points to the time series. The data is then divided into equal-sized blocks related to the synchronous signal being provided by the tachometer, and these blocks are averaged together. The current block would start on the leading edge of the tachometer signal and end on the corresponding point preceding the following tachometer pulse. By taking sufficient averages, an improved estimate of the desired signal can be determined since any random noise will be eliminated [90].

Data from both machines were sampled intermittently in bursts, a fixed length time block, for our experiments. The amount of time per burst is called the burst duration or burst length. Intermittent sampling reduces the volume of vibration data that needs to be processed and/or stored, and consequently, overcomes storage space restrictions and minimizes communication overhead which are known issues accompanying the management of continuous streaming data [38].

Like a turbine, the rotation of the blades in a box fan result in vibration patterns that are representative of the state of the fan. To record its vibration, two accelerometers were attached at different angles to the enclosure of an ordinary 50 cm box fan. The accelerometers used were both low noise, low frequency accelerometers (model AC136-1A) with a sensitivity of 500 mV/g and a low noise power spectral density (PSD) of $1.7 (\mu g)^2/\text{Hz}$ at 10Hz. The first accelerometer, represented by Channel 1, was mounted to the fan using a mounting pad. The second accelerometer, Channel 2, was glued to the surface of the fan. The sensor data were sampled at a rate of 1,000

Hz (one thousand readings per second).

Upon completion of the dynamometer, six single-axis accelerometers – 2 high frequency AC104 accelerometers and four low frequency AC136-1A accelerometers – were mounted in different places on the dynamometer to allow for the acquisition of raw vibration data. The locations of these sensors are shown in Figure 2.4. In our experiments, we refer to these accelerometers as channels. The four low-frequency accelerometers, AC_LF1, AC_LF2, AC_LF3 and AC_LF4, are located closest to the prime mover and are channels 4, 3, 2, and 1 respectively. AC_HF1 (channel 6) and AC_HF2 (channel 5) are the high frequency vibration sensors which record the vibration from more rapid rotations by GENX. All sensors except the AC_HF1 sensor at channel 6 were installed at or around 90 degrees relative to axial view with prime mover in the foreground (i.e., MIMOSA Convention). AC_HF1 was placed at 180 degrees. The sampling rate for these sensors was 5,000 Hz.

A low-pass filter at 2,000 Hz removes aliasing which would have otherwise made sensor signals indistinguishable from each other at frequencies greater than half the sampling frequency. A high-pass filter at 0.1 Hz removes the effects of DC frequencies.

4.2 DATA MANIPULATION/TRANSFORMATION & FEATURE EXTRACTION

Once the vibration data is collected from the accelerometers, it needs to be transformed to find the frequencies with the greatest amount of signal. Unlike some sensor types (oil, temperature, leak, etc.), it is not sufficient to examine the latest state of the accelerometer. Vibrations are composed of waves, with each frequency taking a different amount of time to register and providing different information about the health of the dynamometer. Transformations from the time-displacement domain

(the raw waveform data) to other domains including the frequency-amplitude domain (showing the degree to which different frequencies contribute to the signal) and the time-frequency-amplitude domain (a three dimensional plot showing which frequencies contribute to the signal and how the relative contribution of these frequencies changes over time) are used to extract information useful for classification. The time-frequency-amplitude domain is difficult to produce using Fourier-based methods, as these require choosing a fixed-width time window in advance for analysis. Hence, we chose to employ wavelet transforms, an increasingly popular technique in machine condition monitoring [118, 141].

Wavelet analysis provides distinct advantages over other preprocessing methods such as Cepstrum analysis and Fourier transforms, and is highly efficient on streaming data. Unlike Fourier Transforms, wavelet analysis works on a multi-scale basis (on both time and frequency domains), is suitable for non-stationary signals, and has varying window sizes which allows for higher frequency bands to be sampled more frequently and lower frequency bands less often. Thus, by using wavelet transforms, we are able to preserve the time domain and allow for analysis of possible patterns within each burst.

As the name suggests, wavelet transforms are based on wavelets, functions which exhibit wave-like properties but which are only non-zero for a small portion of their domain (hence wavelet, “little wave”). The analyst first chooses a so-called “mother wavelet,” and then creates a family of “child wavelets” by sliding (translating) and scaling (dilating) the mother wavelet. Each of these child wavelets is individually multiplied with the original signal, showing how much of that signal can be explained by oscillations of that child wavelet’s translation and dilation. These values, representing time and scale respectively, can be assembled into a scalogram which shows how much of the original signal is explained by each scale for each point in time.

Due to the computational complexities of creating a large number of child wavelets and multiplying each with the signal, a simplified form of wavelet transformation known as the discrete wavelet transform was introduced. This employs a “father wavelet”, similar to a low-pass filter, in addition to the mother wavelet (which acts as a band-pass filter). In the first round of transformation, both the mother and father wavelets are applied separately to the signal, producing two vectors: a detail vector which contains information found at the current level of resolution (produced by the mother wavelet) and another, the approximation vector, which contains all information found at lower resolutions than the current one (produced by the father wavelet). These wavelets are then recursively applied to the approximation vector from the previous level, as seen in Figure 4.1 (e.g., the approximation vector from level 1 is used to produce the approximation and detail vectors on level 2, then the approximation vector from level 2 is used to produce the approximation and detail vectors on level 3, etc.). Note that in this figure, $g[n]$ represents the father wavelet and $h[n]$ represents the mother wavelet.

The $\downarrow 2$ functions indicate that each vector contains only half as many values as the vector used to produce it. The output of the wavelet transformation is a collection of detail vectors from each scale (level), as well as the final approximation vector from the lowest level of the transform. (The extraneous approximation vectors, although essential to calculating the wavelet transform, do not constitute part of the output.) As with the traditional wavelet transform, these can be combined to create a scalogram.

4.2.1 Streaming Wavelet Transform

One downside of the traditional wavelet transform approaches is that they require all data to be present before transformation can begin. This is not always the case,

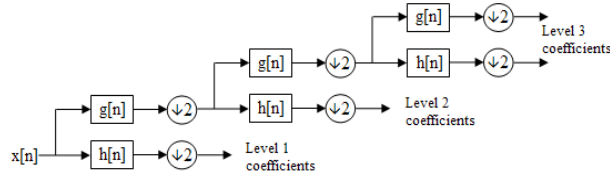


Figure 4.1: Wavelet Transform Filter Bank (Credit: Wikimedia Foundation, Inc.)

especially for streaming data where new information is arriving from a live sensor and must be transformed and processed in real time. For situations such as this, a streaming wavelet transform (SWT) is needed. Such an algorithm was proposed in [152], with its implementation used in these experiments. This takes an approach similar to the discrete wavelet transform, but lazily evaluates each vector, only using the information which has been seen so far. As new input becomes available, the algorithm checks if there is enough information to produce values for the deeper levels of the transform, and calculates these only when the new information is ready. To gain additional efficiencies, each vector only stores the most recent value calculated for that level and vector type (approximation or detail); when a new value is found, it replaces the old one. Overall, the output of the SWT will be a collection of vectors, one for every other time instance (due to the pairing nature of the transform, adjacent time instances have identical values and only one of each pair need be reported in the output). In other words, there are half as many data points in the output as there were in the input. Each vector contains the latest values for all levels of the transform which were available at that time.

One side effect of this SWT algorithm is that for the earliest time instances, some of the values within their output vectors were not computed, thus resulting in missing values. For example, for the first fifteen time instances, there is not enough data to compute the fourth level of decomposition (i.e. the fourth data value in the

output vector) since sixteen time instances have not yet elapsed. So for all of those instances, the fourth level of decomposition is missing. Overall, the first $2^{k-1} - 1$ instances will have no values for depth level k of the decomposition. Thus, the output of this algorithm has relatively large number of missing values, and some attributes (in particular, the ones for larger values of k representing low frequencies) will have more missing values than others. These are not noise or error, but a known consequence of the algorithm, and classification must take this into account. This seeming shortcoming could be remedied by continuous streaming rather than – as in these experiments – sampling in bursts. The problem with that, however, would be the software and bandwidth limitations when processing waveform data streamed at 5,000 Hz from eventually twelve sensors (six phasor measurement sensors for power quality in addition to the six accelerometers).

4.2.2 Short Time Wavelet Transform

The Short Time Wavelet Transform with Baseline-Differencing (STWTB) methodology, which is fully defined in Wald *et. al* [153], is a second technique (used in our more recent work) to derive features from the raw vibration signals. Through further experimentation as presented in the referenced publication, it was determined that the applied wavelet transforms did not produce the necessary information to allow learners to build models robust enough to detect problems if the turbine’s current operational speed is different from the speed at which the data used to build the models was gathered. This motivated the change to the STWTB approach.

The STWTB is a two step process where the first step is the application of a Short Time Wavelet Transformation (STWT) algorithm which converts the time series of amplitude readings into a time-frequency representation of the signal. The second step is the use of “baseline-differencing” to normalize the data relative to a given operating

condition. Like the SWT described above, the STWT algorithm implemented as a part of the STWTB uses a Haar wavelet – a simple but well-known wavelet [137, 152]. Each instance or observation in the file output by this algorithm is a vector of n features where each feature says how much of the original signal can be represented by oscillations at a given wavelength.

The second step in the STWTB process is to generate a baseline from the data gathered during a specific “normal” operational condition, and then to subtract the baseline from the current observations made by that source regardless of the current operational conditions of the machine. This baseline-differencing step was deemed necessary to remove those portions of the vibration signals that are characteristic of the machine’s environment and/or operational conditions (in this case, its rotational velocity), so that the remaining signal only depicts the vibrations caused by actual abnormalities in the machine (and not its operating conditions).

4.3 FUSION TECHNIQUES

In our experiments, we employ three data fusion techniques, one from each fusion level defined in Section 3.3. These techniques are described in greater detail below.

4.3.1 Data-Level Fusion

In order to combine sensor signals at the data level, they must have originated from sources which produce the same type of signal [40]. Our case studies employ a simplified data level fusion algorithm which takes the arithmetic mean of corresponding data values between each sensor channel. This was performed on the raw accelerometer data (*i.e.*, prior to wavelet transformation).

4.3.2 Feature level fusion

Feature level fusion [42] involves first extracting features or attributes which describe the data, and then combining the features from each signal to produce a fused signal. To fuse the sensor data at the feature level, a set union of the features produced by the wavelet transform from all channels was performed, which, intuitively, should improve a classifier's ability to perform state detection since all the available data is being taken into account during the data mining process.

For a given experimental setup and a set $S = \{S_1, S_2, \dots, S_m\}$ of sensor sources, n features are extracted from each source S_i . We will use S_{ij} to denote the j^{th} feature extracted from the data from source S_i . The fused output from all m sources is given by:

$$F(S) = \bigcup_{j=1}^n \bigcup_{i=1}^m S_{ij}$$

Simply put, the output of the fusion process will be a file containing all the features from all sources, or a total of $n \times m$ features. For simplicity, each source is considered equally reliable and thus its observations are considered no more or less important than any of the other sources.

4.3.3 Decision level fusion

Decision level fusion [36] involves making a local decision from each signal and then combining the decisions to get the final output. This can be done by combining the decisions themselves or by combining the probabilities of class membership for each class and selecting the class with the highest probability [149]. We use the latter approach.

Given the set $S = \{S_1, S_2, S_3, \dots, S_n\}$ where n is the number of sources and s_i is the observation of the i^{th} source as well as the set of possible classes $C = \{c_1, \dots, c_m\}$,

a model d_i is generated from the observations in the training set made by source S_i using a predetermined machine learning algorithm. We will therefore have n models, one generated from each source’s observation.

Each model d_i is then applied to each instance j in the test set. Each model produces *measurement* level details about the predicted class of j , meaning that it also outputs the probability of instance j belonging to each class c_k . For each instance, we have $n \times m$ probabilities: the probability given by each of n models for each of m classes. The overall probability that instance j belongs to class c_k is computed by averaging the probabilities given by all n models for that class. The class with the highest average probability is selected to be the predicted state for instance j .

The classifiers generated on each subset of data could originate from the same machine learning algorithm, as is done here, or from a combination of different learning algorithms, which can be done in a future experiment. Learners used for our experiments are described next.

4.4 CLASSIFIERS

The WEKA (Waikato Environment for Knowledge Analysis) software package¹ is an open source collection of data mining and machine learning algorithms [67]. Its graphical user interface allows for ease of use and the vast array of data mining related tasks it supports makes it a popular choice for researchers in the field. These tasks include data preprocessing (e.g., data imputation), classification, regression (useful for prognostics) and feature selection. The nine machine learning algorithms used in our studies are all implemented in the WEKA workbench. These are:

1. C4.5 Decision Tree (C4.5O and C4.5N): The Decision Tree is a tree-like ma-

¹Available on <http://www.cs.waikato.ac.nz/ml/weka>

chine learning model. Decision rules comprised of comparisons of attributes to numeric thresholds are coded as branches, and the predicted values (which in a classification problem would be the class labels) are coded as leaves. The C4.5 algorithm [122] generates decision trees recursively, computing each of its comparison thresholds based on the information gain (i.e., the difference in entropy) of the independent attributes at each level. We built two decision tree classifiers using J48, the WEKA implementation of the C4.5 decision tree algorithm. The first one was constructed using default parameters (labeled C4.5O); for the second, denoted C4.5N, we disabled pruning, enabled Laplace smoothing and used default values for remaining parameters.

2. Naive Bayes (NB): A simplified form of a Bayesian network, the Naive Bayes learner applies Bayes' rule of conditional probability and an assumption of independence among the features to predict the probability that an instance belongs to a specific class. Previous studies have demonstrated the effectiveness of this classifier even when dependencies exist among the features [26].
3. Multi-Layer Perceptron (MLP): The Multi-Layer Perceptron (MLP) neural network [14] is a form of feed-forward neural network which maps input values to an output. It uses a learning technique known as back-propagation, a generalization of the least mean squares algorithm, which involves continually updating the weights it assigns to individual connections within the neural network based on the amount of error in the output compared to the expected outcome. Two parameters were changed: the *hiddenLayers* parameter (the number of nodes in the intermediate, or hidden, layers in the network) was set to '3' and the *validationSetSize* (percentage of the training dataset reserved for validating the MLP model during back-propagation) was set to '10'.

4. Repeated Incremental Pruning to Produce Error Reduction (RIPPER): The RIPPER algorithm [21], an inductive rule learner, performs classification of new instances based on a set of rules generated from a learning dataset. Rule generation involves greedily adding antecedents based on information gain and then pruning the resulting rule to optimize the rule set and minimize errors. Our experiments used the default parameters for the JRip algorithm, which is the WEKA implementation of RIPPER.
5. k -Nearest Neighbor algorithm (2NN and 5NN): In the k -Nearest Neighbors algorithm (abbreviated kNN), the predicted class of an instance is determined by a majority vote of the k closest training examples in the feature space, where the distance is defined in terms of the feature space. In the WEKA implementation, called IBk, the default search algorithm for the kNN algorithm is a brute force algorithm which finds the examples with the shortest Euclidean distance. Default values were selected for all parameters of the IBk algorithm with the exception of the value of k which was set to 5 for the 5NN classifier and 2 for the 2NN classifier.
6. Support Vector Machine (SVM): The simplest form of the Support Vector Machine (SVM) is a hyperplane which divides a set of instances into two classes with maximum margin. Its support vectors are a subset of instances which are used to find this hyperplane. While there are many forms of SVMs, WEKA implements John Platt's sequential minimal optimization (SMO) algorithm [121] which uses analytic quadratic programming to solve the Lagrange multipliers and a heuristic function for determining which multipliers are to be optimized. Two parameters for SMO algorithm were changed: c (representing the complexity constant of the SVM) was set to 5.0 and *buildLogisticModels* (which allows

the SVM to obtain proper probability estimates) was set to ‘true’.

7. Random Forest (RF10 and RF100): The Random Forest (RF) classifier [7] is an ensemble of randomly generated unpruned decision trees. The output of this classifier is the mode of the decision of the individual trees, each trained based on a random selection of features and a random sampling of instances. The classifier using default values is denoted RF10 while the one constructed when the parameter for the number of trees is set to 100 is denoted as RF100.
8. Radial Basis Function Neural Network (RBF): A Radial Basis Function Network (RBF) [9] is a simple neural network containing a single layer of nodes where the activation function is a radial basis function. In WEKA, the RBF is implemented as a normalized Gaussian radial basis function network, which learns logistic regression over a k -means clustering algorithm to provide the basis functions. No changes were made to the default WEKA parameter values for this learner.
9. Logistic Regression (LR): In standard logistic regression (LR), maximum likelihood estimation is used to perform binary classification based on the calculated probability of having a given output as a function of the values of the attributes. The probabilities are based on a parametric model having parameters estimated from the training data. The WEKA implementation uses a multinomial logistic regression model, while allow multiple classes, with a ridge estimator [89] to improve parameter estimates and minimize error. Default WEKA parameters were used.

Parameter values were tuned in accordance with findings from previous studies [3, 157].

4.5 FEATURE SELECTION

Feature selection techniques aim to reduce the dimensionality of the feature space by removing unnecessary features. We apply these techniques in some of our feature-level fusion experiments to reduce the quantity of data that the classifiers would need to process. Our feature selection experiment(s) employed eight different techniques – described below – based on results of previous experiments conducted by our team [3, 157] on different datasets. Of these eight, the first two techniques are available in the WEKA tool. The remaining six were proposed and implemented in the framework of the WEKA tool by our team.

- The Chi-squared method (χ^2 , or CS) [19] – based on the χ^2 -statistic – ranks the features according to their statistical dependency to that class. A feature is considered to be more relevant to a class for larger Chi-squared values.
- The Information Gain (IG) [3] technique determines the relevance of a feature by calculating the amount of information that is gained about a class when that feature is used.
- The ReliefF algorithm [61] estimates the rank of a feature based on how well its values distinguish between instances (or rows of data) if they are from different classes while showing no difference if the instances are of the same class.
- The Mutual Information (MI) is a measure of the mutual dependence between two random variables; lower values of the mutual information statistic indicate a greater degree of independence [157].
- The use of Kolmogorov-Smirnov (KS) as a feature ranking/selection technique was first proposed in [84] by a member of our team. This method works by

using the KS statistic to measure the maximum differences between the empirical distribution function of the attribute values of instances in each class. A feature is said to be better able to distinguish between two classes for larger distances between the calculated distribution functions. Our case study is its first application as a feature selection technique in the reliability analysis context.

- The Deviance (Dev) is a measure of the residual sum of the squared errors according to some pre-defined threshold. Smaller values of deviance are preferred since it is representative of the degree of error.
- Also used as a performance measure, the AUC [71] is a single numeric value representing the area under the Receiver Operating Characteristic (ROC) curve (TPR versus FPR). AUC values range from 0 to 1, where a value of 1 indicates perfect performance.
- The feature ranking method based on the area under the Precision-Recall Curve (PRC), which is also called the average precision, is similar in nature to the one based on the AUC. The main difference here is the use of the PRC (graph of PPV against TPR) instead of the ROC.
- The Signal-To-Noise (S2N) [3] is a less popular technique. It ranks features according to how well each feature discriminates between two classes. For each feature, the S2N of that feature is calculated by dividing the difference between the mean value of the features from the positive and negative classes by the sum of the standard deviations of that feature for both classes.

4.6 DATA IMPUTATION

To remove the missingness factor in the streaming wavelet transform data, we consider two dissimilar approaches – Expectation-Maximation Imputation (EMI) and Mean Imputation (MI).

The EMI algorithm [140] is a popular tool for statistically estimating values that are missing in the input dataset. It is a deterministic iterative algorithm which determines the maximum likelihood estimates of the parameters of the distribution which the complete (missing and observed) data are assumed to follow. It starts with an initial guess of the value and performs two steps (the expectation and maximization steps) during each iteration; after several repetitions, the algorithm converges. In the first step, the expectation step, the conditional expectation of the log-likelihood evaluated using the current estimate for the parameters is computed. The maximization step involves computing new estimates for the parameters by maximizing the expected log-likelihood from the expectation step. These parameter-estimates are then used to determine the distribution of the latent variables in the next expectation step.

MI is quite simple compared to EMI. In MI, missing values are replaced by the mean value of the non-missing values of that attribute for that class. Its simplicity makes this algorithm quite fast, but typically less effective than EMI because of its tendency to under-estimate the sample variance [83].

4.7 PERFORMANCE MEASURES

In a binary classification problem, a classification model is trained to output one of two possible values or classes based on the input values for that instance in the dataset. The class of interest (the abnormal state) is dubbed the positive class while

the normal class is considered the negative class. The number of instances that are correctly identified by a classifier as being positive are the true positives (TP) and the number of negative instances that the classifier labeled as positive are the false positives (FP). Conversely, the true negatives (TN) are the number of instances correctly identified as being normal or not faulty while the false negative (FN) count is the number of incorrectly labeled positive instances – positive instances labeled incorrectly as negative. Given the total number of positive instances P and the total number of negative instances N, the true positive rate (TPR), false positive rate (FPR) and accuracy (ACC) can be computed from the TP, FP and TN counts as follows:

$$\begin{aligned}
TPR &= \frac{TP}{P} \\
FPR &= \frac{FP}{N} \\
Accuracy &= \frac{TN+TP}{P+N}
\end{aligned} \tag{4.1}$$

Four performance measures used in the experiments are the False Positive Rate ($FPR = \frac{FP}{N}$), the False Negative Rate ($FNR = \frac{FN}{P}$), accuracy and the area under the ROC curve (AUC). The ROC (Receiver Operating Characteristic) curve is a graph of the False Positive Rate on the x -axis versus the True Positive Rate ($TPR = 1 - FNR$) on the y -axis. Each point on the curve represents the TPR and FPR at a different decision threshold. The decision threshold is used by classifiers when assigning a class label to a new instance; if the confidence in labeling an instance as being faulty or abnormal exceeds this threshold then the instance is labeled as faulty. Lowering this threshold increases the TP and decreases the TN, making the classifier more biased to the faulty (or positive) class. Varying the threshold, therefore, affects the FPR and FNR and the optimal value of this parameter depends on the class distribution and the data itself.

A single, numeric value ranging from 0 to 1, the AUC summarizes the information contained within the ROC curve and combines information about how well a classifier performs on both the positive and negative classes. Perfect classification models produce an AUC of 1. For high assurance systems such as the ocean turbine, an ideal state detection module must maximize this value, and even small differences in this value could mean a significant loss or gain in performance and in turn, the reliability of the condition monitoring system.

4.8 PERFORMANCE EVALUATION

Once a classification model is built from a set of training data, its performance can be evaluated in one of two ways: using n -fold cross validation or using a test set. The former approach evaluates the model on data similar in nature to the data used to build the model; the latter allows the researcher to test the possibility that certain characteristics of the data may have changed since training. Earlier experiments [32, 40, 42, 45, 43, 48] evaluate models use some form of n -fold cross validation while our more recent work [33, 35, 37, 36, 47] employ a training set.

In n -fold cross validation, the dataset is randomly split into n subsets. Subsets 1 through $n - 1$ are used as the training set and the last subset as a test set. Each model is built on the training set and evaluated on the test set. This process is repeated by using each of the n subsets exactly once as the test set, and using the remaining subsets each time as the training set. The average result across all n iterations is computed as the output.

For feature selection experiments, we implemented n -fold cross validation with feature selection as a methodology for selecting the subset of p features and training the classification models. Instead of selecting features from the entire dataset, feature

selection is done on the $n-1$ training folds for each iteration during the cross validation process. So, for each of the n iterations, a subset of features are re-computed on the $n-1$ folds of training data using the feature selection technique. Once these p features are selected, a classification model is built using this reduced set of features from the $n-1$ folds of training data and then evaluated on the remaining fold that was put aside for testing.

Data and Knowledge Fusion Framework for MCM/PHM in Inaccessible Ocean Systems

5.1 INTRODUCTION

Research on data fusion techniques, architectures and approaches spans many domains including medical diagnosis [162], military defense and tracking systems [136], robotics [115], navigation systems for autonomous vehicles [22] and remote sensing [13]. To the author's knowledge, however, little work had been done in data fusion for MCM/PHM in unattended, inaccessible¹ systems such as ocean turbines. The majority of research effort invested in data fusion for autonomous systems (whether on land or sea) has been related to designing vehicle navigational systems. Our research is, therefore, not only unique in the target system (ocean turbines) but also in the intuition behind a mathematical data fusion model for MCM/PHM systems that it provides.

5.2 CHAPTER CONTRIBUTIONS

The main contributions of this chapter are:

¹With a goal of a one year maintenance free operation with annual preventative maintenance

1. A data fusion approach to MCM/PHM systems whose architecture satisfies the OSA-CBM specification.
2. A formalization of barrier synchronization as a technique for coordinating sensor data streams prior to fusion
3. A case study showing preliminary results of simple data fusion performed on vibration data

The OSA-CBM specification was described in Section 2.2. In Section 5.4, we will present a data fusion model which can be applied to MCM systems designed based on the ISO-13374 standard. We will apply this model to a case study in Section 5.5, present results of preliminary analysis in Section 5.6, and summarize our findings in Section 5.7.

5.3 RELATED WORK

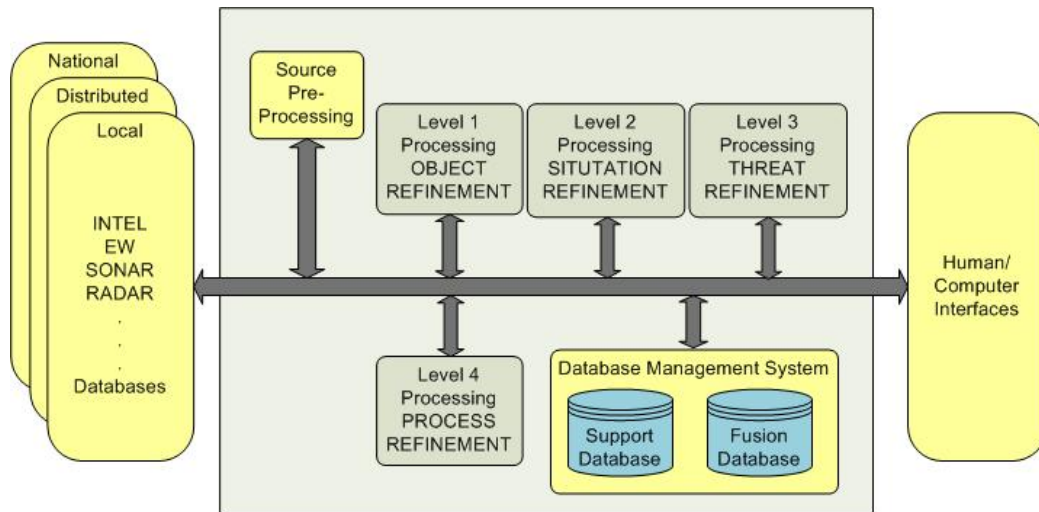


Figure 5.1: JDL Process Model for Data Fusion

5.3.1 Data Fusion

Data Fusion is considered a cross cutting concern of an MCM/PHM system because the data to be integrated can be provided by several entities and layers in the OSA-CBM architecture [123]. To streamline and standardize the design and codification of data fusion systems, experts in the area have proposed more than 30 fusion architectures over the years. Some of these models are discussed in [50]. Of these, one of the most widely cited model is that of the American Joint Directors of Laboratories, or JDL [136], which was initially developed for military applications. The JDL model [138] divides the processes, functions and techniques applicable to data fusion into five levels (as seen in Figure 5.2). These are:

Level 0. Pre-Processing – A level 0 data preparation/estimation process estimates entity features from one or more entity signal observations [99].

Level 1. Object Assessment – In this phase, an attempt is made to locate and/or identify the object of interest by fusing information about this object which was gathered from multiple sources. The object assessment level is itself broken down into four sub-steps, namely:

1. Data Alignment – Data processing occurs to align the data into a common frame of reference (e.g. spatial or temporal).
2. Data Association – Relationships among data points are identified. For example, in surveillance systems, a data association function would attempt to group all the measurements from different platforms (that is, any object that is carrying a sensor) which are associated with the same target. In an MCM/PHM system, oil, temperature and vibration measurements could be associated with the component they are measuring.

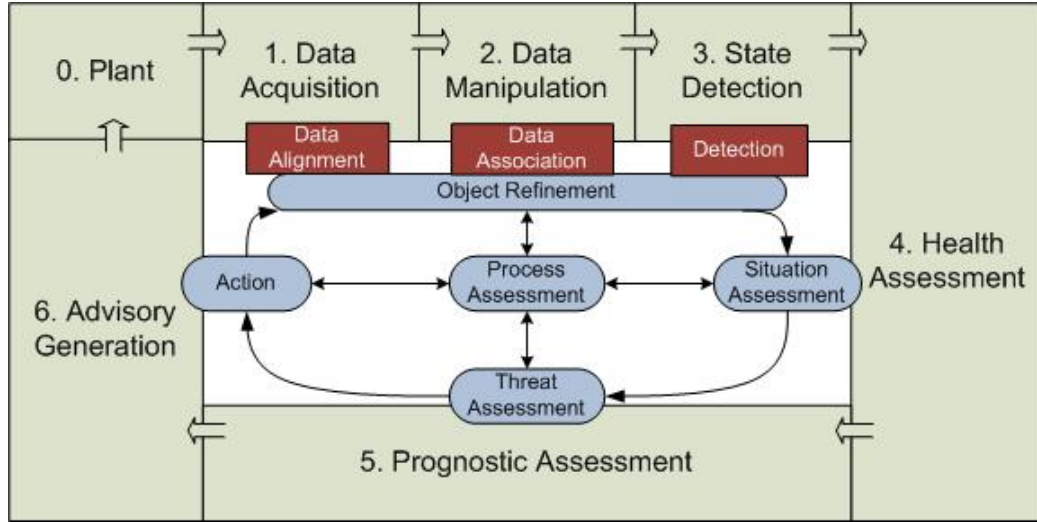


Figure 5.2: Data Fusion and MCM Model

3. State Estimation – The target’s state is calculated from the measurements obtained from the previous levels.
4. Identification – An attempt is made to predict the identity or classification of an object.

Level 2. Situation Assessment (SA) – The results from the previous level are interpreted to establish a relationship between the reconstructed entity and an observed event [50].

Level 3. Threat Assessment – In the threat assessment stage, the outcomes of different plans to remedy the situation are analyzed and the best course of action is predicted.

Level 4. Process Assessment – The process assessment phase is a global level in which the effectiveness and performance of the overall process (both hardware and software) are reviewed to identify possible means of improving the system. This phase in the JDL process involves planning and control.

5.4 FRAMEWORK

Knowing where and when fusion should be performed is considered a primary issue of data fusion [66]. In this section, we will relate the JDL data fusion model to the OSA-CBM model and present a unified perspective on data fusion within the context of MCM/PHM systems for ocean turbines.

Our fusion model, shown in Figure 5.2, presents the data fusion process of the JDL model as an overlay on the layers in the OSA-CBM system architecture to indicate how and where fusion is occurring within the MCM/PHM system. By identifying at what points in the OSA-CBM architecture data fusion is occurring, we are addressing data fusion as a cross cutting concern.

A Plant block has been added to the MCM/PHM architecture to represent the actual hardware portion of the system including its sensors and relays. This Plant will produce the raw data which will be used to drive this PHM system.

The proposed model was adopted from the JDL model which has been slightly modified to include both a Detection phase and an Action phase (as shown in the diagram). The Detection phase replaces the State Estimation and Identification steps in the JDL Model. Since there is no clear-cut distinction between these two sub-steps of the Object Refinement phase for MCM/PHM systems, these steps have been combined to form the Detection phase. In this Detection phase, any anomalies in data will be identified and associated with the component which generated that fault. A new phase – Action – has been added to correspond to Advisory Generation. In this phase, advisories from all the other phases will be consolidated and interpreted to determine if and how the machine can self-adjust to optimize its current state.

The Process Assessment phase remains at a global level and can retrieve data from and provide feedback to other phases in the data fusion model. This phase needs to

be able to measure the efficacy of the data fusion process and trigger any adjustments (for example, to fusion parameters) to ensure optimal performance while satisfying system requirements. One researcher pointed out the inherent difficulty in evaluating a data fusion process in terms of the intended objectives [66], thus creating the need for a viable evaluation technique that can be used within the Process Assessment phase. This remains an open issue.

As discussed in Section 5.3.1, data fusion can occur at many different levels in the MCM/PHM architecture. Determining the type of data fusion algorithm suitable for each fusion level requires an analysis of the type of sensor data available as well as the type of inference that is desired. In other words, a fusion algorithm that would compute the system state based on the feedback from multiple sensors and output the system state as either “operational” or “malfunctioning” would differ from one needed to coordinate and integrate two homogeneous waveforms.

5.5 CASE STUDY: OCEAN TURBINE MCM/PHM

Inaccessible ocean machinery such as ocean turbines are highly intolerant to false alarms due to the high costs of equipment retrieval. Also, fault detection must occur within a reasonable time-frame to minimize damage to the turbine. An MCM/PHM system for ocean turbines must, therefore, satisfy response time requirements and minimize occurrence of false alarms and silent failures. Data fusion is useful in meeting these requirements as it will allow for more efficient fault diagnosis. We will discuss detailed treatment of silent failures or false negatives in future work.

Because of the continuous nature of the data flow from some of the sensors used within the Plant of the MCM/PHM system, the data fusion sub-system must be capable of handling real time data streaming from multiple, usually heterogeneous,

sensors. In this section, we will apply the data fusion model presented in Section 5.4 and discuss the application of data fusion to an MCM/PHM system for ocean turbines.

5.5.1 Applying the Framework

The Plant block of the model shown in Figure 5.2 represents the hardware components within the MCM/PHM system. In an ocean turbine, this Plant block consists of sensors and relays. Some examples of sensors that can be included within this block are accelerometers, tachometers, oil sensors, pressure sensors, leak sensors, thermometers, atmospheric pressure sensors and electrical output sensors.

Within the Data Acquisition (DA) block of the architecture, data is acquired for different physical phenomena such as vibration, temperature and oil quality. The first level of data fusion (as will be discussed in Section 5.5.3) can be implemented here to accomplish data alignment.

While the model of Figure 5.2 indicates when data fusion is occurring, it provides no notion of how fusion will occur. Zooming down to the data fusion processes which are occurring at the different levels in the MCM/PHM architecture, we identified 3 unique types of data fusion that can occur within this architecture. These are labeled F1, F2 and F3 in Figures 5.4 and 5.5, and will be discussed in Sections 5.5.3, 5.5.4 and 5.5.5 respectively. From discussion in this section, it is clear that a single fusion process will occur across multiple levels of the hybrid framework proposed in Section 5.4.

Before sensor streams can be fused, we need to ensure that the data items being fused are properly coordinated. A formalization of barrier synchronization for coordination of timed sensor data streams relevant to this domain is given in Section 5.5.2.

5.5.2 Barrier Synchronization

Given two sensors or sources within an MCM/PHM system, it often happens that signals being sent to a central processing system are received at different times although they were generated simultaneously. Synchronization between sensors within a data fusion system is necessary to ensure that the data fusion is occurring only between corresponding data points. To do so, the time component of all data points being fused must be aligned with each other within some reasonably sized margin ϵ . This can be accomplished through implementation of a coordination paradigm used in networking, and distributed and parallel computing – barrier synchronization [145].

In terms of vibration sensors, the term *packets* may refer to individual bursts of data. For other sensors, these packets could be the independent batches of readings being transmitted from that sensor to the topside system performing the data fusion. Thus, it cannot be assumed that the packet sizes of the different data streams are uniform. Here, we will propose a barrier synchronization scheme for continuous timed data streams – one in which incoming data items arrive continuously and consecutively [69] – from sensors in a distributed network. Let ϵ be some small, non-negative predefined integer which is provided as a parameter to the synchronization process. A larger value of ϵ will allow for more lenient synchronization and lower wait times while a smaller value of ϵ would ensure a tighter correspondence between data items before fusion. ϵ is one such parameter which can be manually or automatically tuned from within the Process Assessment phase of the Data Fusion Model.

The barrier synchronization scheme is represented diagrammatically by Figure 5.3, which shows the barrier synchronization of the p^{th} timed data item from packet j in data stream s with the q^{th} timed data item from packet k in data stream u .

An example of a timed data item in this domain is a single amplitude reading

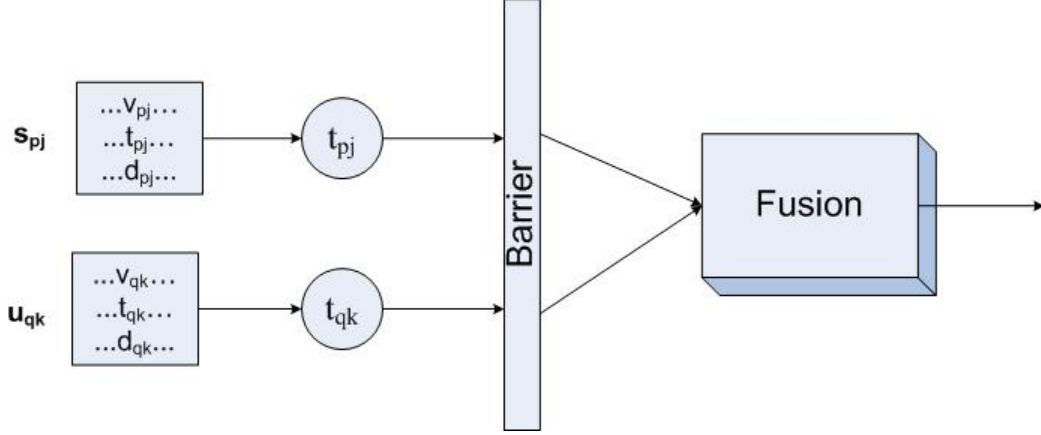


Figure 5.3: Barrier synchronization of timed data streams

within a burst of vibration data. We will define the barrier synchronization of these two data streams as follows: A timed data stream s consists of a sequence of packets of possibly varying lengths where the length of any packet s_j is denoted by $L(s_j)$ and $s = \{s_0, s_1, \dots, s_j, s_{j+1}, s_{j+2}, \dots\}$. Any data packet s_j within stream s consists of $L(s_j)$ data items which are each represented as the triple: $s_{pj} = (v, t, d)$ where v is the data component of the p^{th} data item, t is the start time, and d is the duration.

In some cases, such as for vibration data and other bursty data, the d component for all data items within a burst will be uniform since the sampling rate will be constant at least until the end of the burst. So for any two data items s_{pj} and s_{qj} within the same burst/package s_j , $d_{pj} = d_{qj}$ where p and q represent the indices of the two data items and $p \neq q$.

The duration of a data item d will also aid in the detection of missing data within a packet since the start time of the next data item should always be equal to the start time of the current data item plus its duration, and for the detection of lost packets since the sum of the durations of all data items within the current packet plus the sum of the smallest start time and the delay between packets, if any, should equal the start time of the first data item within the next packet.

Assuming that all data items in a packet are sorted in ascending order according to their start time, and all packets within a stream are ordered in ascending order according to the smallest data item start time within that packet, there are three possible scenarios:

1. Data items s_{pj} and u_{qk} arrived at the same time or within a reasonable timeframe of each other, i.e. $t_{pj} \geq t_{qk} - \epsilon$ and $t_{pj} \leq t_{qk} + \epsilon$. In this case, both packets will be permitted to cross the barrier and begin the fusion process.
2. Data item s_{pj} arrived some time before data item u_{qk} , i.e. $t_{pj} < t_{qk} - \epsilon$. This could mean that the corresponding data item from stream u was lost or delayed, and would require the data item s_{pj} to block at the barrier until time $t_{qk} - \epsilon$.
3. Data item s_{pj} arrived some time after data item u_{qk} , i.e. $t_{pj} > t_{qk} + \epsilon$. As in Case 2, data item s_{pj} will block until the arrival of the next data item in packet u_k or until the arrival of the first data item of the next packet u_{k+1} .

In cases 2 and 3, indefinite blocking is possible and could result in high latency and late fault detection since faulty data from the monitoring system will be delayed behind the blocked data item. An alternate and more feasible approach could be to employ known data imputation techniques [85, 147] to create a replacement data item for fusion whenever needed. Another alternative is to introduce a second parameter to the synchronization process, w , which represents the maximum allowed blocking time. If a data item blocks for greater than w time units, the known data point is allowed to pass through the barrier, enter the fusion process as the sole input for that time block and then exit the fusion process as its output. Analysis of an appropriate approach to cases 2 and 3 will be left for future studies.

5.5.3 Inter-Sensor Data Fusion

The first type of fusion, which occurs at the boxes labeled F1 in Figure 5.4, works closest to the raw data emanating from the sensors. At the lowest level, sensor data from two heterogeneous sources will be combined to validate signals, create features, and generate a unified output. In our context, we will be combining vibration signals from the accelerometer through channel C1 with velocity data from the tachometer through channel C3 to produce time synchronous average vibration features [114] through channel C1.3. Inter-sensor data fusion would occur during the Data Alignment phase of our hybrid framework (Figure 2) where the common frame of reference would be the tachometer signal.

In the inter-sensor data fusion process, an analogue-to-digital converter (ADC) is used to first digitize the raw analogue signal that is received from the sensors into a sequence of digital values via a two step process. Sampling, the first step of the digitization process, is necessary to produce a signal in terms of discrete time and continuous values, while quantization, the second step, maps the set of continuous data values to a discrete set.

Once the data has been digitized, a low pass filter could be applied to de-noise the signal and improve the signal-to-noise ratio. Other de-noising techniques include a statistical multichannel filtering approach [124], wavelet transforms, Wiener filter [159] and linear filters.

After undergoing a de-noising process, the vibration signal will then need to be normalized (or *ordered*) with respect to the tachometer through channel C3 via Time Synchronous Averaging (TSA) which was defined in Chapter 4. TSA is a signal processing technique often used to perform inter-sensor data fusion and reduce noise. By normalizing the accelerometer data with respect to the tachometer signal, we

ensure that time samples are aligned to the same angular position of the component (*e.g.* a gear or shaft) to preserve the phase relationship. Additionally, since the rotational speed is not constant, order tracking is needed to distinguish non-stationary vibration data from transient vibrations.

On the opposite side of the component, a second accelerometer measuring the vibration of the same component from a different angular position produces signals through channel C2. The entire fusion process F1 will be mirrored there to normalize the accelerometer signal through channel C2 with respect to the tachometer signal through channel C3 to generate signals through channel C2.3.

5.5.4 Intra-Component Data Fusion

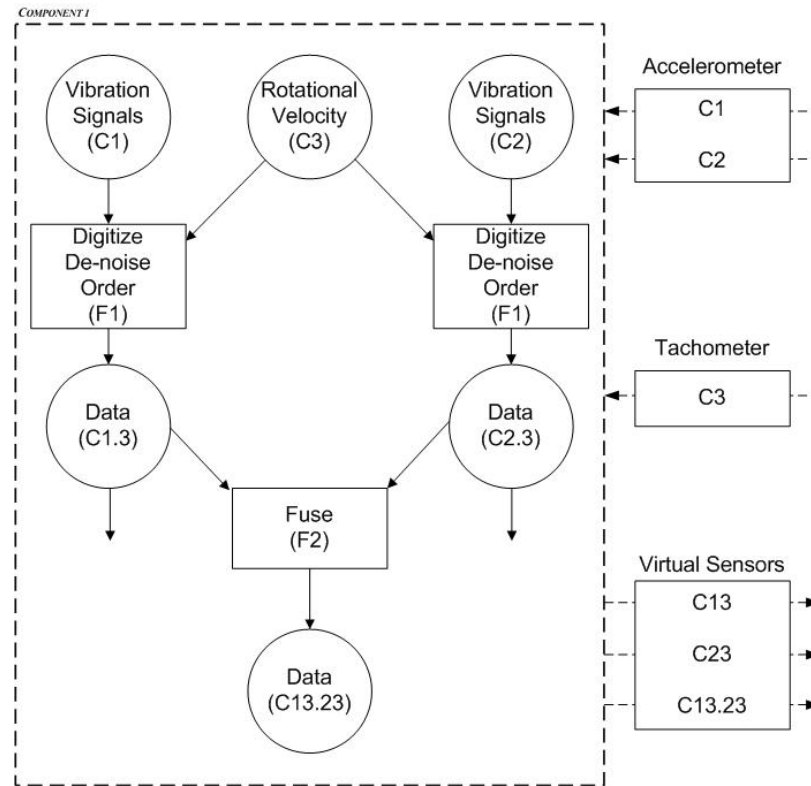


Figure 5.4: Intra-component fusion diagram

In the second type of data fusion, denoted by F2, homogeneous data from multiple sources (channels C1.3 and C2.3) will be integrated to produce more accurate data than could have otherwise been generated from a single sensor. This may be between sensors measuring the same object but at different angular positions or between redundant sensors measuring the same object at the same angular position. In the former case, data from both sensors will need to be included in the final result since each sensor provides a unique perspective of the object. In the latter case, the data will help to identify sensor malfunction and would provide a mechanism for sensor validation. In either case, the fusion process F2 will output data through channel C13.23 in Figure 3. F2 will be used, for example, in the Data Association phase of the Fusion Model to combine vibration readings from multiple accelerometers attached to the same component.

Intra-component sensor fusion can be accomplished by any of three different levels of abstraction: signal/data-level fusion, feature-level fusion, or decision-level fusion. At the data-level, the data fusion algorithm is executed on the raw signals. Greater accuracy is obtained from doing so but this approach is only viable when the signals are of the same type (as would be the case with channels C1.3 and C2.3). At the feature level, feature extraction is done to produce a feature vector on each signal and the extracted feature vectors are fused. A sample feature set would be 320 bins of vibration data that correspond to different frequency ranges in a spectrogram. At the highest level of abstraction (decision-level fusion), each sensor processes their signals independently to produce a local estimate. These local estimates are then combined via a fusion process. By performing decision level fusion, features can be added or removed from the system without having to change the method of analysis. While this does provide a significant advantage, accuracy is sacrificed. Data level fusion will be performed here to maximize accuracy.

Techniques for data-level, feature-level and decision-level intra-component fusion will be investigated in future studies.

5.5.5 Inter-Component Data Fusion

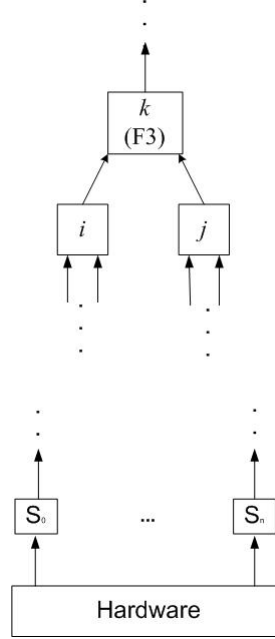


Figure 5.5: Inter-component fusion diagram

The health of a complex system cannot truly be determined from individual models and data analysis techniques. So, at our final level, we combine data from multiple components to enable system-level diagnosis and prognostics. This level of fusion, referred to as knowledge fusion, involves integrating higher-order, possibly competing data and will be observed only in upper level nodes in the system architecture. Figure 5.5 diagrammatically represents this concept. Note that fusion types F1 and F2 will only occur within components i and j if they are lower level components (meaning that the input to these components are raw sensor signals), but F3 would occur within any other component to fuse the virtual sensor signals from any two compo-

nents below it in the architecture. Inter-component fusion would occur during the Situation Assessment phase of Figures 1 and 2 as it establishes how a fault in a single component will affect the health and life expectancy of the entire system, particularly adjacent components.

We now need to consider how to not only combine heterogeneous data from multiple sources, but how the fusion process will operate in the presence of discordance. Early research done in psychology on knowledge fusion within humans and animals resulted in the definition of four possible ways to integrate input from different types of components [70]. The four modes, known as Bower's Taxonomy of Fusion, are:

1. *complete unity*, wherein the data from all the components are combined without any consideration for the possible discord, creating the illusion of compromise between sensors. No system is in place for detecting any disagreements;
2. *unity with awareness of discordance and the possibility of recalibration*, in which the conflicts are detectable and are reconciled by recalibrating the offending sensors;
3. *unity with awareness of discordance and tendency towards suppression*, in which disagreements are detectable but the information gathered from the offending sensors is temporarily ignored;
4. *no unity at all*, where there is no combination of the sensor data and no way to detect conflicts. Because of its triviality, this mode will be omitted from any future discussion.

In complex environments, the appropriate mode of integration may depend on the situation. It is also not feasible to apply the same mode of integration throughout the entire system since the system will need to adapt to changing environmental

and operational conditions [144]. So, an approach to knowledge fusion must consider which mode of fusion is appropriate to the current situation.

Because of the ability to incorporate human knowledge in determining the precedence of signals in the event of the conflict, an expert system could be built to distinguish the appropriate mode for a specific situation and perform inter-component fusion. Liu and Liu previously implemented an expert system via a fuzzy group multiple attribute decision making method to perform this level of fusion in a machine condition monitoring system [95]. In their approach, the expert system was comprised of four modules, namely the diagnosis tree, a fuzzy group multiple attribute decision maker, a knowledge base in which each rule was associated with a confidence factor representing the level of uncertainty in the validity of the rule, and an inference engine.

Future work may include the use of expert systems to accomplish this level of fusion.

5.6 EXPERIMENTAL SETUP

The following experiment was conducted to analyze simple intra-component fusion, F2, on real vibration data by investigating its effect on state detection from vibration signals. Although the fusion algorithm used for F2 is trivial, these results will show if and how state detection is improved or impaired by a fusion algorithm.

To simulate the vibration of an ocean turbine as the propeller rotates, two accelerometers were attached at different angles to the enclosure of an ordinary 50 cm box fan. The first accelerometer, represented by Channel 1 (C1), was mounted to the fan using a mounting pad. The second accelerometer, Channel 2 (or C2), was glued to the surface of the fan. For this simple experiment, the fan and the two sensors

are considered our hardware plant for our MCM/PHM system. The accelerometers were both connected to a WaveBook Data Acquisition Unit, which was responsible for both data acquisition (DA) and time synchronous averaging of the vibration data.

Vibration measurements were then sampled at a frequency of 1000 Hz, or 1000 readings per second, in three (3) second bursts for each of the following four experimental setups:

1. The fan placed on a solid floor, operating normally and in upright position at 1010 revolutions per minute (rpm). This scenario presents a baseline (BL)
2. The fan tilted on a soft surface such as the operator's hand (TOH)
3. The fan tilted on a hard surface such as a wall (TOW)
4. The running fan slowed with an obstruction, like a pencil or popsicle stick (SWO)

All experiments were run independently and the faults (i.e. TOH, TOW and SWO) were not introduced while the fan was running normally. Each experiment was run a total of six (6) times with equal burst durations and sampling rates, producing a balanced class distribution which, although uncharacteristic of data produced by a live turbine, is adequate for initial tests and vibration analysis.

Corresponding data points from channels C1 and C2 were averaged to simulate a simple data-level fusion process F2. The resultant fused signal is similar to that of virtual channel C13.23 in Figure 5.4, and is referred to as Channel 3 (C3) within this experiment.

The streaming Haar wavelet transform (SWT), described in Section 4.2.1, was then used to convert the resultant time series of amplitude readings into a time series of resonance vectors, mapping real valued amplitudes to a tractably small set

of distinct resolution scales. For the wavelet transform used in this case study, we examined wavelengths from 2^1 to 2^{10} and consequently, the output of the transform contained ten attributes, one for each scale 1 through 10, whose value would be 1 if a wave is detected at that scale or 0 if no matching wave is found.

Next, the time frequency features produced by the wavelet transform for each of the three channels are passed through a sliding window transform (with window sizes of 50, 100 and 200) to reduce short term variations in the signal. To implement a sliding window transform with a window size of w , we sum the values of the next w instances for each of the ten attributes. The value of each of the ten attributes would therefore be between 0 and w . The instances containing the summed values are combined to form a new dataset, which is used in place of the original dataset.

The six runs for each experiment were merged to form a single dataset for each of the four classes (BL, SWO, TOH and TOW). For our binary classification problem, we then combined all six runs of the BL class with each of the three classes of interest SWO, TOH, and TOW producing three sets of data each having six runs of the class of interest plus six runs of baseline data. Therefore, in total, for this experiment, there were $3 \text{ window sizes} \times 3 \text{ classes of interest} \times 3 \text{ channels} = 27 \text{ datasets}$.

Twelve data mining classifiers were then trained on each of the 27 datasets to distinguish between normal operation (BL) and the single class of interest (TOH, TOW or SWO). Each classifier was built using the WEKA data mining tool, version 3.5.2; default parameters were used unless otherwise noted. In the results, we assign a letter label (letters A through L) to each classifier to save space in the graphs in the results section. We employed: both versions of the C4.5 decision tree (A-C4.5O and J-C4.5N), Naive Bayes (B-NB), Multi-Layer Perceptron (C-MLP), Repeated Incremental Pruning to Produce Error Reduction (D-RIPPER), both versions of the k -Nearest Neighbor algorithm (K-2NN and E-5NN), Support Vector Machine (F-

SVM), both versions of the Random Forest (L-RF10 and G-RF100), Radial Basis Function Neural Network (H-RBF) and the Logistic Regression (I-LR). Details on these classifiers were presented in Section 4.4.

Each classifier was built using one run of 5-fold cross validation (CV). In 5-fold CV, the dataset described above is randomly partitioned into five parts, called folds, with the first four folds serving as a training set and the last fold held back as a test set. The cross validation process is repeated four more times such that each fold serves as a test set once, and then the results of all five folds are averaged to produce a single estimate.

The performances of these algorithms were expressed in terms of the AUC – the area under the ROC curve – which was also defined in Chapter 4.

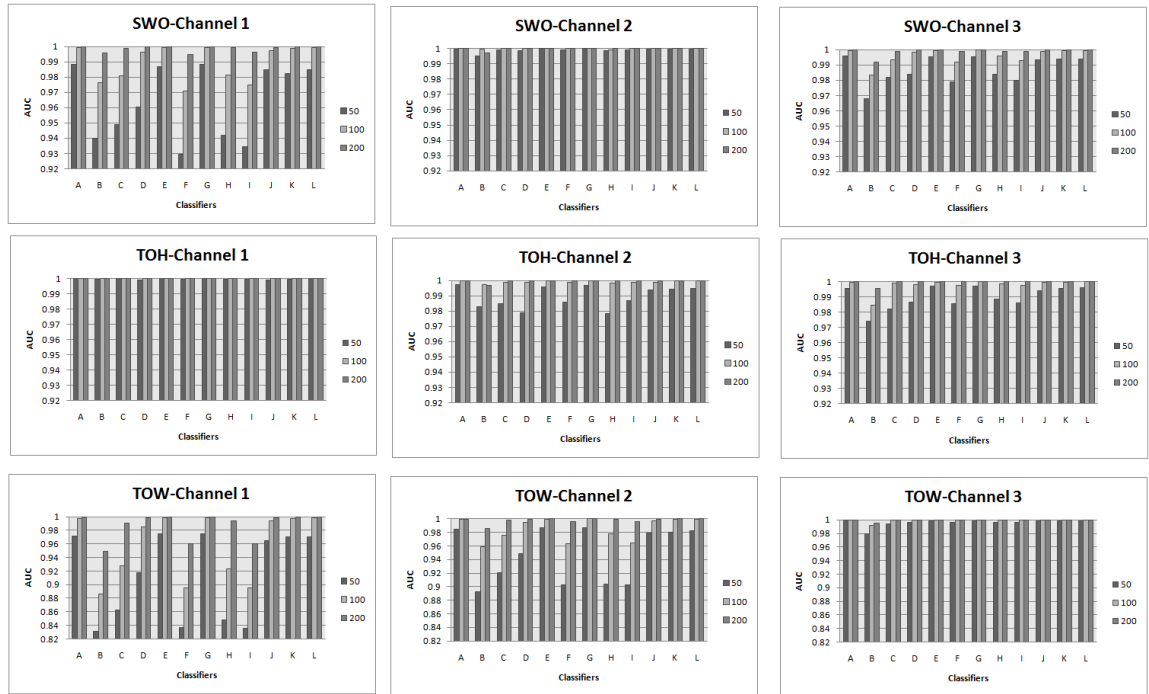


Figure 5.6: Results

5.6.1 Results

In this section, we present the classification performance of each of the twelve classifiers on the experimental data. From these results, we answer the following research questions:

1. Are multiple sensors needed to measure the vibration of the same component?
2. Does data fusion improve classification performance?
3. What is the effect of window size on classifier performance?
4. Which classifiers are best and least able to detect a state of interest from the wavelet transformed vibration data?

Figure 5.6 shows the AUC values for learners A through L on each of the three classes of interest – slowed with obstruction (SWO), tilted on hand (TOH) and tilted on wall (TOW) – and for three channels: C1, C2, and C3. Each class of interest is presented in a different row; SWO is shown in the first row, TOH in the middle row and TOW in the last row. Results for channels C1, C2 and C3 are displayed in the first, second and third columns respectively. Each clustered bar graph shows the AUC values on the y-axis for three window sizes (50, 100 and 200) and each of the twelve classifiers for a specified class of interest.

First, we will look at the results for channels C1 and C2 to see if classifiers were able to better detect the state of interest from any one channel. C1 and C2 results suggest that classifiers are better able to distinguish SWO and TOW from C2 data but perform better on C1 data for recognizing TOH. So, by removing any of the two channels, we may be sacrificing the ability of the health monitoring system to detect one or more states. This demonstrates the importance of having multiple

accelerometers measuring the vibration of the same component from different angular positions, and establishes the need for some type of fusion algorithm to aggregate the data from these sensors.

To determine whether data fusion enhances state detection by improving classification performance, we will compare the three graphs of the channel C3 results to those from the other two channels. For SWO, all twelve classifiers achieved a minimum 0.99 AUC from C2 data regardless of window size, but showed greatly degraded performance on C1 data across all three window sizes. All classifiers were better able to detect SWO from C3 than from C1, but performed best on C2 data. All classifiers were able to achieve an almost perfect AUC when detecting TOH from C1 data and at least a 0.97 AUC when detecting TOH from C2 data. Although classifier detection of TOH did not improve after fusion when compared to any single channel, the AUC values achieved on C3 were within a 0.01 AUC of those from C2. For TOW, all but the top six classifiers had degraded performance on C1 data. Although detection of TOW on C2 data was slightly better than on C1 data, the best performances on TOW data (for all twelve classifiers) regardless of window sizes were recorded for the fused channel C3, where AUCs exceeded 0.98 across the board. So, even in a most naive and oversimplified form, data fusion provided more stable classifier performance than data from any single channel.

Our results also showed the effect of window size on classifier performance. The smallest window size needed to guarantee a minimum AUC of 0.98 was 200 on both C1 for the top nine classifiers and C2 for all classifiers. By applying simple data fusion and a window size of just 100 instances, we observe the same (or better) performance as from a window size of 200 instances on C1 or C2 individually. So, from our fused channel C3, a window size of 100 would be sufficient.

From the results, we can also determine the best classifiers for state detection for vibration analysis of rotating machinery. Our results indicate that the five classifiers that were best able to distinguish the class of interest (TOH, TOW or SWO) from normal operation (BL) were G-RF100, L-RF10, E-5NN, K-2NN, A-C4.5O, while the worst performing classifier was B-NB. These rankings were consistent across all three channels. Overall, the RF classifier proved to be the most robust and consistent classifier in this study. These results concur with preliminary empirical studies [54] [82], which investigated the robustness of the RF algorithm for classification of noisy and/or imbalanced data.

5.7 CHAPTER SUMMARY

This chapter defines a framework and approach to data fusion for MCM/PHM systems in inaccessible, unattended ocean systems. The proposed data fusion approach has been partitioned into three levels: inter-sensor or fusion of data between heterogeneous sensors, intra-component or data fusion between homogeneous sensors, and inter-component fusion which is performed at higher levels of the architecture to provide an overview of the health of the system. At the inter-component and intra-component levels of fusion, the performance and error rates of distributed classification and expert systems must be carefully measured to determine the effectiveness of these techniques.

The results of the experiments discussed within this chapter showed that data fusion, even in a most naive and oversimplified form, provided more stable classifier performance and higher correlations between runs than data from any single channel. In the upcoming chapters, we will analyze other data fusion approaches and will consider data mining pre-processing techniques such as feature extraction where

necessary.

Preliminary Analysis of Data Mining & Sensor Fusion Techniques on Fan Data

6.1 INTRODUCTION

In this chapter, we evaluate and compare the three fusion techniques and the twelve machine learning algorithms on combining and learning from homogeneous data acquired from two vibration sensors mounted on a box fan. Each of the three fusion approaches is representative of a different level of fusion. The techniques are applied at different points in time after data acquisition: immediately after the raw data has been gathered (data-level fusion), after features have been extracted from the data (feature-level fusion), and after a decision has been made regarding the state of the machine (decision-level fusion). The three fusion levels, the fusion techniques and the machine learners were previously defined in Section 3.3.

In the two experiments described in this chapter, we implement the top three blocks of the OSA-CBM architecture (refer to Chapter 2.2 for more details) and build a simple MCM/PHM system for rotating machinery. In these studies, data acquisition (DA) is done via a WaveBook Data Acquisition Unit, which is also used within the data manipulation (DM) block to perform time synchronous averaging (TSA). In addition to TSA, our DM block involves the Streaming Wavelet Transform (SWT) –

explained in Section 4.2.1 – for feature extraction and signal processing. Data mining and machine learning techniques provide an avenue for automated interpretation of the sensor data and problem classification, while sensor fusion techniques are needed to combine data from multiple sources to get a complete, more accurate picture. These techniques would work collaboratively within the SD and HA blocks to allow for complete, tested and automatic interpretation of the raw data, and identify problem states as they occur.

The experiments discussed in Section 6.2 analyze and compare the performance of data-level, feature-level and decision-level fusion algorithms for intra-component fusion of vibration data. For both experiments, we focus on distinguishing between normal operation and a single faulty state (referred to as binary classification), thus the output of our SD block can only be one of two values – normal or faulty. These experiments demonstrate homogeneous sensor fusion of vibration data and demonstrate the abilities of the three fusion levels to improve the ability of machine learning algorithms to detect the orientation and operating state of a turbine. Detection of multiple faulty states, or multi-class classification, has been left for future work.

6.1.1 Contributions

To the authors’ knowledge, little work has been done in comparing these levels of sensor fusion based on analysis of experimental data or on studying sensor fusion within an MCM/PHM system. The few studies comparing the performance of these three fusion levels were specific to different application domains [128]. This research is therefore not only unique in its domain, but in its use of experimental data to rank the suitability of these three fusion approaches for this domain. Results presented here provide useful insight into the performances of the various machine learning algorithms and fusion techniques on distinguishing system state based on vibration

data preprocessed by a wavelet transform and lay the foundation for the experiments presented in the subsequent chapters.

The remainder of the chapter is laid out as follows. Section 6.2 follows with a description of the experiments conducted, the results of which are discussed in Section 6.3. A summary of our findings and chapter conclusions are provided in Section 6.4.

6.2 EXPERIMENTAL SETUP

These experiments compare different fusion and mining approaches for fusing and interpreting vibration data gathered from a typical 50 cm 120V AC box fan. As the researchers did not yet have access to vibration data from the ocean turbine at the time this study was conducted, two identical accelerometers (model AC136-1A) were glued to the outer casing of the box fan and readings were taken while the fan was running in various states. A turbine is basically a large fan whose blades rotate at approximately 60 RPMs. Although a fan runs at a much higher speed than a turbine, the rotation of its blades produces different vibration signatures depending on varying operating conditions, as would a turbine.

An IO-Tech Wavebook/516-E Data Acquisition Unit (DAQ) was used to record the sensor data and perform TSA [90]. An important note is that the measurements recorded from both accelerometers during these experiments were already synchronized (i.e., ordered) and complete with no data points missing. Also, there was no class imbalance, i.e. the number of data points which were normal was the same as the number of faulty data points. Also, data imputation techniques [147, 48] may also be utilized to fill in data which may have been lost during transmission and/or because of sensor malfunction. Class imbalance and data imputation are considered

in later experiments.

6.2.1 Fan Experiment 1

For the first experiment (EXP1), measurements were recorded from the fan while it was operating at 1010 RPMs in four different setups: standing upright (baseline), tilted on a soft surface i.e. a hand (TOH), tilted on a hard surface i.e. a wall (TOW), and slowed with an object i.e. a pencil (SWO). These four experiments correlate to four possible scenarios while an ocean turbine is submerged – running normally, tilted on its mooring line, tilted on a submerged object like a piling and obstructed by debris. The baseline state is considered the normal class, while TOH, TOW and SWO are the faults or problem scenarios that the data mining classifiers will try to detect.

For each setup, data from the two accelerometers were sampled at 1000 Hz in 3 second bursts, producing a total of 3000 readings per burst. These accelerometers are denoted as channels 1 and 2 (CH1 and CH2). Each experiment was repeated a total of six (6) times, resulting in 18,000 measurements per experiment. The six runs for each setup were combined resulting in 8 files = 2 channels x 4 setups.

6.2.2 Fan Experiment 2

The second experiment (EXP2) was conducted from the same two accelerometers on the same fan, operating at 420 RPMs. Aside from a baseline state in which the fan was allowed to run upright without obstruction, three faulty states were produced by slightly percussing the side of the fan (PP), using the tip of a pencil to slow the fan blades (SO), and pushing the same pencil further into the blades so that its distal portions make contact with the fan thus producing a hard obstruction (HO). The PP setup simulates the contacts of sharks to the turbine casing, while SO and HO relate

to the turbine blades being obstructed by debris.

Data from both accelerometers (CH1 and CH2) were sampled at 1000 Hz for 1 second for each setup, with each setup repeated four times. There were, therefore, 1000 readings per burst and 4000 readings per setup. The data set for this experiment consisted of 8 files = 2 channels x 4 setups.

6.2.3 Pre-Processing and Fusion

Data level fusion (FT1) was performed on the raw accelerometer data by computing the arithmetic mean of corresponding data points. That is, $FT1(t) = \frac{CH1(t)+CH2(t)}{2}$. FT1 was applied on the data from each of the four setups and each experiment. The number of files after FT1 is 12 files = (2 channels x 4 setups) + (1 fused channel x 4 setups) for each experiment.

The data in all 24 files (12 files from each experiment) were passed through the streaming wavelet transform (SWT) described in Chapter 4.2.1. For our purpose, the transform is used to detect oscillations or vibration signatures. To find these oscillations, the transform observes the transitions between pairs of data points at a time, and outputs 1 if an oscillation is detected at that scale and time and 0 otherwise. The transform, therefore, produces one data point for every pair of data points it examines, resulting in output files having half the number of lines as the input, which reduces the amount of data that needs to be processed and/or stored.

This algorithm is simple, computationally fast and was found to be quite effective in previous work [42]. The number of wavelet features produced by the transform is equal to the number of resolution scales that the transform will examine, where scale n corresponds to a wavelength of 2^n . The number of scales is a parameter to the wavelet transform process. For the 3000 data points contained within each file in EXP1, wavelengths greater than 1024 (2^{10}) would not be meaningful as they would

only produce 1 value for the entire burst. 2^9 was selected as the largest wavelength for EXP2 since it had only 1000 data points per file, and a wavelength of 1024 would be too large. Therefore, the output of the wavelet transform on EXP1 and EXP2 had 10 and 9 binary features respectively, where the values for each feature could be either 0 or 1.

After performing the wavelet transform and extracting the features from each of the data files in each experiment, feature level fusion (FT2) was performed by taking a union of all of the wavelet features from CH1 and CH2 for each experiment and each setup. Let $CH1_i(t)$ represent the i^{th} feature at time t for CH1. $FT2(t) = \{CH1_1(t), CH1_2(t), \dots, CH1_n(t)\} \cup \{CH2_1(t), CH2_2(t), \dots, CH2_n(t)\}$ where n is the number of features. So, for EXP1, the 10 wavelet features from CH1 were fused via process FT2 with the 10 wavelet features from CH2 for each setup, producing 4 new files, each containing 20 features. Similarly, for EXP2, the 9 wavelet features for corresponding data points from each channel were combined. Counting the 8 files from the individual channels, the 4 FT1 files and the 4 new files with the combined features (FT2), EXP1 and EXP2 both have 16 data files = (2 channels x 4 setups) + (4 setups with FT1) + (4 setups with FT2).

A sliding window transform [91] with a window size of 100 was then applied to the transformed data in each of the 16 data files for each experiment to reduce short term variations in the signal. Initial experimentation described in the previous chapter revealed that windowing improved the performance of data mining learners on this data and that a window size of 100 was sufficient to do so. This transform was implemented as a sum of the values for each feature over the length of the window. To implement a sliding window transform with a window size of w , we sum the values of the next w instances for each of the ten attributes; the value of each of the ten attributes would therefore be between 0 and w .

The resultant sliding window transformed data files are used in place of the original files for training twelve data mining learners. In a binary classification problem, the goal is to distinguish between two states or classes. For our experiments, we intend to distinguish a single faulty state from baseline, so we combined the four data files for the BL setup (i.e., the baseline files for CH1, CH2, FT1 and FT2) with each of the three faulty classes for the corresponding channel. BL represents our normal state, while the faulty state is our class of interest. For EXP1, these faulty states are SWO, TOH and TOW; for EXP2, the classes of interest are PP, SO and HO. Note that the BL data file for EXP1 is different from the BL data file for EXP2. After combining the files for classification, we have, for each experiment, a total of 12 datasets = 3 classes of interest \times {CH1, CH2, FT1, FT2}.

Twelve data mining classifiers – see Section 4.4 – were then trained on each of the 12 datasets to distinguish between BL and the class of interest. These classifiers are available in the WEKA [67] data mining software package¹. Each classifier was built using one run of 5-fold cross validation. In 5-fold cross validation, each data file is randomly partitioned into five parts, or folds, with the first four folds used to train the data mining algorithm and the last fold held back as a test set. The cross validation process is repeated four more times such that each fold serves as a test set once. The results of all five folds are averaged to produce a single estimate.

For each instance within the data files, in addition to the predicted class, the learners output the probability of belonging to each class. Decision level fusion (FT3) was performed on the classification results of CH1 and CH2 by averaging the probabilities of membership for each of the two classes for that instance and selecting the predicted class with the larger probability. Say, for example, that for a given data point in the CH1 data file at time t , the probability of being faulty according to CH1 data is 0.65

¹Available on <http://www.cs.waikato.ac.nz/ml/weka>

while the probability that the CH2 data point occurring at time t is faulty is 0.15. The probability of being faulty for data point $FT3(t)$ is $\frac{0.15+0.65}{2} = 0.40$. Since there are only two classes, the probability of being normal for that data point is $1 - 0.40 = 0.60$. Since that data point has a higher probability for being normal, its predicted class is BL, or normal.

The performances of each of the learners on each channel (CH1, CH2, FT1, FT2, FT3) were measured in terms of the AUC, which is defined in Section 4.7. Results are shown in Section 6.3.

6.3 EMPIRICAL RESULTS

In this section, we present the AUC values for each of the twelve classifiers on the different setups and channels for the two experiments. These results, shown in Figures 6.1 and 6.2 for EXP1 and EXP2 respectively, will be used to compare our three fusion approaches – data-level fusion (FT1), feature-level fusion (FT2) and decision-level fusion (FT3) – to the results from individual channels (CH1 and CH2). We hope to gather not only which level of fusion yields the best results, but also how consistent these results are across multiple experiments. Both figures follow the same general format.

In each figure, there are three graphs – one for each setup or fault within that experiment. These graphs each contain twelve clusters of five vertical bars, where each vertical bar represents a different channel and each cluster is a classifier. From left to right, the five vertical bars represent results for CH1, CH2, FT1, FT2 and FT3. The twelve clusters reading from the left are the C4.5 decision tree with tuned parameters (C4.5N), Naive Bayes (NB), Multi-Layer Perceptron (MLP), RIPPER (RIP), 5-Nearest Neighbors (5NN), Support Vector Machine (SVM), Random Forest

with 100 trees (RF100), Radial Basis Function Network (RBF), Logistic Regression (LR), C4.5 decision tree with default parameters (C4.5O), 2-Nearest Neighbors (2NN) and Random Forest with 10 trees (RF10). The height of each bar is the AUC value, thus the maximum value is 1.

The scale of each graph has been modified for readability. For example, in the TOH graph in Figure 6.1, the minimum value on the vertical axis is 0.975 compared to 0 for the TOW graph since all of the AUC values for TOH were larger than 0.975.

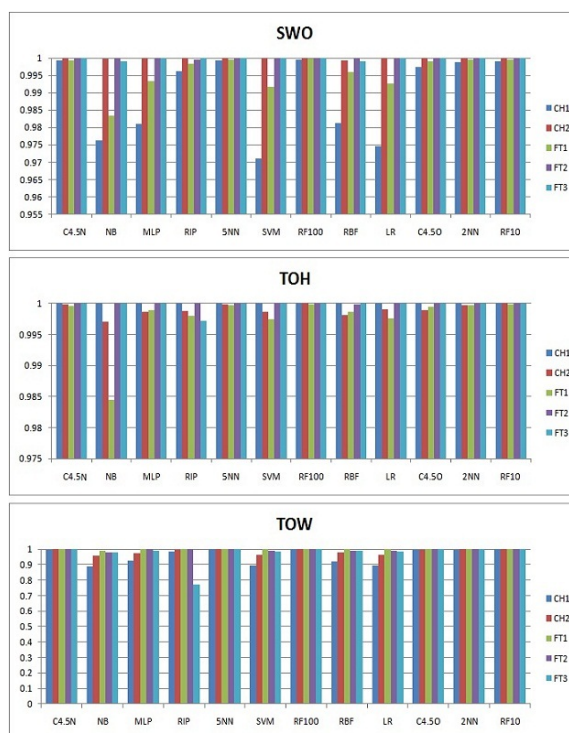


Figure 6.1: Fan Experiment 1 - AUC per Learner and Fusion Technique

6.3.1 Results for Fan Experiment 1

Figure 6.1 shows the results of the first fan experiment (EXP1). Here, the results for the three classes of interest (SWO, TOH, TOW) are presented as separate graphs in that order from top to bottom.

For SWO, the classification performance for all classifiers on feature-level fusion (FT2) and decision-level fusion (FT3) were better than data-level fusion (FT1), where the difference in performance between FT2 and FT3 were within 0.001 AUC of each other. Of all three fusion approaches on SWO data, FT1 performed the worst while FT2 performed the best overall. On TOH data, FT1 performed the worst and FT2 performed the best overall. Also, for TOH, all classifiers achieved an AUC of 1 on CH1 data and a minimum AUC of 0.999 on FT2, and performed better on FT2 and FT3 than on FT1. All three fused channels showed better results than any individual channel (CH1 or CH2) on TOW data, with FT1 performing the worst and FT3 performing the best. The poor performance of FT1 was likely due to its naive implementation as a simple averaging. We will confirm this through experimentation with different data-level algorithms in future work.

For both SWO and TOH, FT2 and FT3 outperformed FT1, while on TOW, FT1 outperformed all other channels. Overall, FT2 seemed to provide the most consistent classification performances of the five channels, while FT3 did not outperform any of the other two fusion channels for any of the three setups.

6.3.2 Results for Fan Experiment 2

The results for the periodic percussion (PP), light obstruction (LO) and hard obstruction (HO) setups for EXP2 are shown in Figure 6.2 from top to bottom.

For the PP setup, all classifiers achieved an AUC of 1 across for all three fused channels. Classifier performance when distinguishing LO was worse than that for PP. While FT1 performed better than CH2 data on LO, FT2 was the best and most stable fusion approach for that setup. On HO data, FT1 provided the worst results of all three fusion channels, while FT2 yielded the most consistent results.

We found that overall, FT2 outperformed individual sensor channels CH1 and

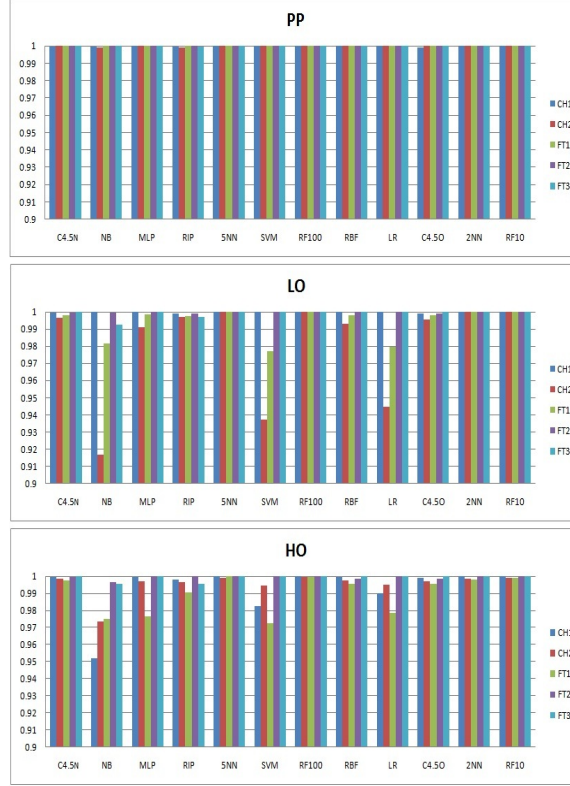


Figure 6.2: Fan Experiment 2 - AUC per Learner and Fusion Technique

CH2. A ranking of the overall performances of the three fusion levels on both experiments in decreasing order would yield: feature-level fusion (FT2), decision-level fusion (FT3), and then naive data-level fusion (FT1).

In terms of the learners, we find that using FT2 or FT3, the C4.5N, RF10 and RF100 models all produce similar results. The 2NN and 5NN models also behave similarly.

6.4 CONCLUSION

In this chapter, we compared the performance differences of twelve data mining classifiers on data fused at the data-level, feature-level and decision-level. Data were collected from two accelerometers mounted on a typical household box fan. Based

on the results from both experiments, we noted that feature-level fusion yielded the most stable results and the greatest improvement overall compared to the other fused channels. Feature level fusion also yielded more consistent results than individual channels in most cases.

In the next chapter, we continue our experiments on data gathered from our test bed (the dynamometer). This provides the opportunity to analyze said techniques on data from a machine that operates more like the turbine than the fan does.

State Detection From Imperfect Data

7.1 INTRODUCTION

In the previous chapters, we conducted experiments on vibration data sampled at 1,000 readings per second from two sensors mounted on a household fan. This was a simplified version of the MCM/PHM problem with which we are faced but preliminary findings from those experiments helped shape the direction of the research presented in this chapter and the next. The data used in all remaining case studies were gathered from the six vibration sensors installed on the dynamometer testbed, which behaves more similarly to the ocean turbine than the house fan, thus producing more realistic datasets. This chapter considers the following:

- Due to the nature of the Streaming Wavelet Transform (SWT) algorithm, at certain points in time, some of the values within the output vectors are not yet computed, thus resulting in missing values. Roughly eleven percent of the values produced by the wavelet transform from bursts of 20,000 instances will be missing.
- Three times more sensors and five times faster sampling means that there is 15 times more data being collected per second from the dynamometer than from the fan. Based on the quantity of data available in the fan experiments, at most

18 features could be extracted (using the wavelet methodologies described in Chapter 4) from both sensor channels; however, using the dynamometer, feature level fusion of the six sensor channels would produce up to 72 wavelet features (12 per channel) if the wavelet transform was applied on just a one second burst of data. With four times more features, we consider feature selection as a means for reducing the quantity of data that a learner would need to process by removing redundant or unnecessary features.

Accelerometers mounted on the dynamometer record the degree of vibration of different parts of the machine. Changes in the underlying pattern of these vibration signals typically relate to some type of event, such as a change in speed, development of faults or the presence of wear or erratic behavior. The streaming wavelet transform algorithm was implemented and used to preprocess data for the first two out of three case studies presented in this chapter. This third case study employed the Short Time Wavelet Transform with Baseline-Differencing (STWTB). As before, classification models were generated within WEKA. Descriptions and parameter settings for the various classifiers were given in Section 4.4 and those for the wavelet algorithms were provided in Section 4.2.

7.2 CONTRIBUTIONS

The main contributions of this chapter are three case studies: the first (Section 7.3) concerns missing data while the second and third (Sections 7.4.1 and 7.4.3) revolve around feature selection. To the authors' knowledge, this research is the only such work which investigates missingness to this extent in the context of reliability analysis, and more specifically, to ocean turbine MCM/PHM. Also, it is the only known work applying feature selection techniques to ocean turbine vibration data and comparing

the performances of feature selection techniques for enabling reliable state detection.

7.3 CASE STUDY 1 – MISSING DATA

Construction of reliable classification models from imperfect data is one of the biggest challenges in domains where data imperfections are prevalent. Incompleteness, which is one form of data imperfection, occurs whenever a source such as a sensor, transducer or human observer provides insufficient information about the phenomenon being observed [44]. This missingness hampers the decision making (i.e., classification) process as decisions are generally dependent on full or complete information [101].

Learning from data containing missing values has been the target of many research efforts both within our team [85, 147] and by other experts [164]. Not many of these studies, however, involve datasets where missingness is a natural characteristic of the data and is not randomly injected into the data. This case study revolves around just that – three sets of data where missing values are a direct consequence of the preprocessing algorithm used to generate the data. Prior to imputing or filling in these missing values, we will utilize known machine learning techniques to build classification models from this data and evaluate their abilities to learn from the data in its natural state. We then generate models after the missing values in the data are imputed using two very different techniques – Mean Imputation and Expectation-Maximization (EMI) Imputation. We perform imputation to discover which learners are affected solely by data missingness. Both techniques were described in Section 4.6.

This case study investigates the robustness of various prominent machine learning algorithms trained on imperfect data to distinguish between operating states of an ocean turbine prototype. The study involves three sets of experimental data, each

gathered from the six accelerometers mounted on the dynamometer test bed. Section 7.3.1 describes these three experiments. Recall from Chapter 2 that sensor channels are numbered relative to prime mover MTRX, with the leftmost sensor being channel 1. This sensor configuration is the same across all three experiments and the sensor to channel correspondence is likewise consistent.

For all experiments, we selected drive-shaft rotational speeds within the range of 52.5 and 73 revolutions per minute (RPM) which were experimentally known to avoid resonant frequencies of the dynamometer. An ocean turbine would typically operate between 30 and 60 RPM, so these speeds represent operation toward maximum rated RPM.

Since we have complete control over sampling and experimental conditions, an equal amount of data was collected for normal *vis-a-vis* faulty states for all experiments, resulting in a balanced class distribution. During a field deployment, the time spent in some failure mode will be less than that spent in normal operation, resulting in data exhibiting class imbalance. Experiments presented in the next chapter concern imbalanced datasets. A need for synchronizing vibration signals to other waveform signals like voltage and phasor measurements on a live turbine may also be required [133]. Missing data values in the raw waveforms are possible due to packet loss during transmission, sensor malfunction or latencies introduced by separate DAQ's operating in parallel. These will require some form of data fusion (i.e., ordering) of waveforms to shaft position. Even so, data imputation techniques [147] might be necessary to fill in these missing data values.

7.3.1 Experimental Design

Experiment A

Eight consecutive bursts (4 seconds each) were sampled from the dynamometer without interruption and with SFTA turning at 64 RPM to develop a baseline waveform (BL). We then introduced the first of two simulated faults by lightly percussing the metal case of the GBXA component with a steel tip ball peen hammer. This hammer was equipped with an accelerometer bound to channel 7 whose signal will be analyzed in future work. Percussion proceeded at approximately once per second. This scenario is denoted as SH, for soft hammer. The second faulty scenario was similar to the first with the exception that the impact of the hammer was harder than in the SH case. We label this case as HH, for heavy hammer. These two scenarios were meant to simulate repeated impact from submerged debris or marine animals, or from bolts tumbling in the drive train. The vibration signatures read during a shark attack, for example, could be similar to those from the SH or HH experiment. We collected eight bursts, totaling 32 seconds of data, from each of the six sensor channels and for each faulty scenario.

Experiment B

The dynamometer was again allowed to run without interruption with SFTA rotating at 52.5 RPM to develop a baseline (BL). After sampling the eight bursts of BL data at that speed, the procedure for acquiring the BL signal was repeated twice – once at 64 RPM and the second time at 73 RPM. The soft hammer (SH) experiment from Experiment A was repeated at each of the three speeds, so that for each speed, we have a pair of corresponding BL and SH signals.

Experiment C

The baseline signal for Experiment C was acquired with zero percent resistive load, as in Experiments A and B, and SFTA turning at 50 RPM. In this experiment, we wish to assess vibrations that result from application of a resistive load of forty percent, which creates counter-torque. This is similar to the phenomenon that could be experienced when powering appliances or on-board electrolysis devices. The scenario where the load was applied is considered the abnormal or red state, while data recorded under no load comprises the BL class.

7.3.2 Pre-Processing

The streaming Haar wavelet transform discussed in Section 4.2.1 was implemented then applied separately to each set of data in all three experiments to convert the time series of amplitude readings (in millivolts) into a time series of resonance vectors. There were $8 \text{ bursts} \times 6 \text{ channels} \times 3 \text{ setups} = 144$ files for Experiment A, $8 \text{ bursts} \times 6 \text{ channels} \times 6 \text{ setups} = 288$ files for Experiment B and $8 \text{ bursts} \times 6 \text{ channels} \times 2 \text{ setups} = 96$ files for Experiment C. Each data file output by the Haar wavelet transform contains 10,000 rows, or half of the original number of rows in each of the files prior to applying the transform. This halving occurs during a down-sampling phase of the wavelet transform, where every other instance in the data file is eliminated or removed at the first level of this transform as shown in Figure 7.1. In this figure, we demonstrate the wavelet transformation of 16 instances and show how wavelet features are extracted from the raw signal. The dark arrows represent the extraction of the detail from the previous level; the dashed arrows represent the approximation from the previous level. Approximation values are shown in the gray columns and detail values (each of which is a feature in the actual output vector of the transform)

are in the white columns.

Instance #	Raw Value	Feature #1 (Detail Lvl 1)	Approx. Level #1	Feature #2 (Detail Lvl 2)	Approx. Level #2	Feature #3 (Detail Lvl 3)	Approx. Level #3	Feature #4 (Detail Lvl 4)	Approx. Level #4
0	-0.011368								
1	-0.005646	-0.00286	-0.00851	?		?		?	
2	0.034486								
3	0.064547	-0.01503	0.049517	-0.02901	0.020505	?		?	
4	0.034791								
5	0.005875	0.014458	0.020333	-0.02901		?		?	
6	0.030061								
7	0.052797	-0.01137	0.041429	-0.01055	0.030881	-0.00519	0.025693	?	
8	0.038682								
9	0.00969	0.014496	0.024186	-0.01055		-0.00519		?	
10	0.034486								
11	0.053941	-0.00973	0.044214	-0.01001	0.0342	-0.00519		?	
12	0.02945								
13	0.024796	0.002327	0.027123	-0.01001		-0.00519		?	
14	0.017853								
15	0.011139	0.003357	0.014496	0.006314	0.020810	0.006695	0.027505	0.000906	0.026599

Figure 7.1: Example of Applying Haar Wavelet Transform Showing Missing Values

In the streaming Haar wavelet transform, we take the first pair of instances, i.e. instances 0 and 1 in the diagram. Recall from Section 4.2.1 that the average of these two values forms the approximation for the first level of the transform ($(-0.011368 + -0.005646)/2 = -0.00851$) while half of the difference ($(-0.011368 - -0.005646)/2 = -0.00286$) is the first level detail. The first level approximation calculated from instances 0 and 1 is then paired with the approximation calculated from the first level of the consecutive pair of instances (instances 2 and 3) and the approximation and detail are evaluated for the second level. This detail value, which can only be calculated after the first four time instances have elapsed, will be the second value in the feature vector for instance 3. Even numbered instances will be discarded since the entire feature vector for instance $2k$ will have the same values as the following

instance $2k + 1$. This process is repeated until all possible calculations using the available data have been made.

Consider the example from Figure 7.1 to observe how the missing values and omitted instances arise. First, note that all even numbered instances have no values for either features or approximations. This is because the Haar wavelet only works on pairs of values: it can only be calculated after every two instances, and because the timestamps start with 0, all even-valued instances do not have data. Also observe how instance 1 has only a single feature, while instances 3 and 5 have two features and instances 7, 9, 11, and 13 have three. This is because for the earlier instances, not enough data has yet been acquired for these deeper features to be calculated. These missing values are denoted in the output file (a sample of which is shown in Figure 7.2) by question marks ('?'). Overall, the first $2^{k-1} - 1$ instances will have a missing value for feature at depth k . This means that some instances are missing many values while others are missing few or none; classification with this data must take into account both the quantity and range of missing values. Finally, note that for the deeper features, adjacent instances often share a feature value. This is because these values are only updated when enough new information has arrived to recalculate that value; prior to that, the existing value is used.

The maximum number of features n that will be produced by the transform when there are m instances in the raw file is calculated by $n = \text{floor}(\log_2 m)$. For each raw file containing 20,000 instances (5000 Hz for 4 seconds), we therefore can get no more than $14 = \text{floor}(\log_2 m)$ features. So, in each row of the output files for Experiments A, B and C, there are 14 wavelet features where each feature refers to one frequency resolution. Since the number of samples are identical for all data sets, the proportion of missing values will also be the same. Out of 140,000 values in each data set, 16,369 or 11 percent of them are missing.

1	-0.00488	?	?	?	?	?	?	?	?	?	?	?	?	?	?
3	0.00248	0.005627	?	?	?	?	?	?	?	?	?	?	?	?	?
5	-0.01755	0.005627	?	?	?	?	?	?	?	?	?	?	?	?	?
7	0.004502	0.003967	0.001688	?	?	?	?	?	?	?	?	?	?	?	?
9	0.000534	0.003967	0.001688	?	?	?	?	?	?	?	?	?	?	?	?
11	0.003968	-0.00633	0.001688	?	?	?	?	?	?	?	?	?	?	?	?
13	-0.00881	-0.00633	0.001688	?	?	?	?	?	?	?	?	?	?	?	?
15	0.005837	0.003472	-0.00233	-0.00101	?	?	?	?	?	?	?	?	?	?	?
17	0.010758	0.003472	-0.00233	-0.00101	?	?	?	?	?	?	?	?	?	?	?
19	0.00103	0.009709	-0.00233	-0.00101	?	?	?	?	?	?	?	?	?	?	?
21	-0.00912	0.009709	-0.00233	-0.00101	?	?	?	?	?	?	?	?	?	?	?
23	0.003243	0.0037	0.002356	-0.00101	?	?	?	?	?	?	?	?	?	?	?

Figure 7.2: Sample Output of Wavelet Transform

7.3.3 Classifiers

The case study involves six widely used classifiers, listed below. Also described below is the approach each classifier takes when a model is generated within WEKA from data with missing values.

- Naive Bayes (NB) – In WEKA, the Naive Bayes classifier simply ignores missing values during training and classification.
- Multi-Layer Perceptron (MLP) – Like NB, the MLP ignores missing values during training and prediction.
- k -Nearest Neighbor (5NN) – The maximum possible distance is assigned for an attribute during distance computation if there is a missing value encountered in at least one of the instances being compared. The 5NN model (where $k = 5$) was used in these study.
- Support Vector Machine (SVM) – In the WEKA implementation, missing values are replaced with the mean (for numeric attributes) or the mode (for nominal

attributes) before building the model.

- Decision Tree (C4.5N & C4.5O) – For this learner, an instance with a missing value at an attribute is split up and fractions of it are sent along each branch proportional to how many instances with observed values went down each branch. For example, if there is a binary split on an attribute and 80% of the observed instances go along the first branch (and thus 20% go down the second branch), then the instance with missing values is added to both sets, having a weight of 0.8 (assuming the starting weight was 1.0) in the one branch and a weight of 0.2 in the other. These fractional weights may be further split if the instance has additional missing values.
- Logistic Regression (LR) – Like the SVM, the WEKA implementation of the Logistic Regression replaces missing values with global means/modes.

7.3.4 Classification

For each experiment, we distinguish between a normal state and a single abnormal state (which may be a fault as in Experiments A and B or just a state of interest as in Experiment C), which is a simplified state detection / fault identification problem in the condition monitoring context. To prepare our datasets for each binary classification experiment, we combine the wavelet output from the normal state with that from the given faulty state for the corresponding channel, burst and experiment. After merging these files, the resultant file will contain 20,000 instances: 10,000 instances from each class (normal and faulty). For Experiment A, we will have 96 such files = $\{SH, HH\} \times \{CH1, CH2, CH3, CH4, CH5, CH6\} \times \{BL\} \times 8$ bursts. There will be 144 files = $3 \text{ speeds} \times \{SH\} \times \{CH1, CH2, CH3, CH4, CH5, CH6\} \times \{BL\} \times 8$ bursts for Experiment B and 48 files = $\{40\%Load\} \times \{CH1, CH2, CH3, CH4, CH5, CH6\} \times$

$\{BL\} \times 8$ bursts for Experiment C. Multi-class classification will be examined in future work.

We used WEKA to train the learners on each data file and to evaluate their ability to correctly identify which of the 20,000 instances in each file is red (abnormal) or green (normal) using the performance measures discussed in the next section.

Ten runs of ten-fold cross validation is used to build and evaluate classification models from each dataset. In each run of 10-fold cross validation, the data file is randomly partitioned into ten parts, or folds, with the first nine folds used to train the data mining algorithm and the last fold held back as a test set. The cross validation process is repeated nine more times such that each fold serves as a test set once. The results of all ten folds are averaged to produce a single estimate for that run. Since this is a random partitioning, classification results may differ slightly if the case study is repeated. To minimize this variance and obtain more reliable results, we perform ten runs of this cross validation process and average the results from all ten runs.

After obtaining these results, we apply each of the data imputation techniques (Mean Imputation and Expectation-Maximization Imputation) to each of the wavelet output files and follow the same steps as before to prepare the datasets for binary classification. We again use ten runs of ten-fold cross validation to build and evaluate the classification models generated by each learner. The results are averaged across these ten runs.

7.3.5 Performance Measures

We will use the False Negative Rate (FNR) and False Positive Rate (FPR) as two of the performance measures in this study [130]. We have omitted the TNR and TPR since they can easily be computed from the FNR and FPR. Given the FNR, the TPR is calculated by subtracting the FNR from 1. Similarly, $TNR = 1 - FPR$.

Considering the balanced distribution of the classes in the study, the accuracy will be included in the results as a third performance measure.

For high assurance systems, such as ocean turbines, an optimal fault detection module must minimize the FNR while maintaining an accuracy of at least 0.99. False negatives are analogous to faults that go undetected, which can result in damage to the turbine and the need for unscheduled maintenance and repair. Classification models which do not meet these requirements are therefore sub-optimal and will be considered inadequate for our needs.

7.3.6 Results

The summarized classification results for each classifier on the unimputed data from Experiments A, B and C are presented in Figures 7.3(a), (b) and (c) and discussed in Sections 7.3.6, 7.3.6 and 7.3.6 respectively. For each experiment, we present a table showing the FPR, FNR and Accuracy for each classification model averaged across all eight bursts. Each table contains three columns per abnormal class, one for each performance measure, and one row per learner – Naive Bayes (NB), multi-layer perceptron (MLP), 5-Nearest Neighbors (5NN), Support Vector Machine (SVM), Logistic Regression (LR) and Decision Tree with tuned and default parameters (C4.5N and C4.5O). Results in each table are sorted by accuracy, with the highest accuracies (best performers) listed first. The performance measures are the values averaged across the eight bursts and six channels. Ideal classification models are those which have a FPR below 0.01 and accuracy above 0.99. Higher accuracies allow for greater confidence in the predictions made by these models. FPR and FNR values below 0.01 and accuracy values above 0.99 are highlighted in gray.

Learner	HH			SH		
	FPR	FNR	Accuracy	FPR	FNR	Accuracy
C4.5	0.00174	0.00212	0.99807	0.00254	0.00226	0.99760
C4.5O	0.00242	0.00235	0.99761	0.00282	0.00278	0.99720
MLP	0.01375	0.01964	0.98331	0.02066	0.01384	0.98275
LR	0.07076	0.07899	0.92512	0.07301	0.07053	0.92823
SVM	0.07091	0.08952	0.91979	0.09045	0.06579	0.92188
NB	0.11903	0.17238	0.85430	0.17226	0.16231	0.83272
5-NN	0.21870	0.18840	0.79645	0.22823	0.16342	0.80418

(a) Experiment A - BL vs HH and BL vs SH

Learner	Low Speed (52.5 RPMs)			Medium Speed (64 RPMs)			High Speed (73 RPMs)		
	FPR	FNR	Accuracy	FPR	FNR	Accuracy	FPR	FNR	Accuracy
C4.5N	0.00190	0.00164	0.99823	0.00292	0.00204	0.99752	0.00225	0.00191	0.99792
C4.5O	0.00226	0.00217	0.99778	0.00319	0.00291	0.99695	0.00264	0.00257	0.99739
MLP	0.00594	0.00785	0.99311	0.01340	0.01259	0.98700	0.00804	0.01164	0.99016
LR	0.05462	0.05516	0.94511	0.07932	0.05922	0.93073	0.08548	0.07113	0.92170
SVM	0.06255	0.06025	0.93860	0.09164	0.06823	0.92007	0.09470	0.07388	0.91571
NB	0.11635	0.09514	0.89425	0.23353	0.17086	0.79780	0.09186	0.16245	0.87285
5-NN	0.18429	0.15592	0.82989	0.19689	0.22428	0.78942	0.17619	0.18902	0.81740

(b) Experiment B - BL vs SH where BL and SH recorded 3 times at different speeds

Learner	FPR	FNR	Accuracy
C4.5N	0.00090	0.00266	0.99822
C4.5O	0.00079	0.00422	0.99750
MLP	0.00432	0.00785	0.99392
NB	0.03533	0.05527	0.95470
LR	0.06640	0.08496	0.92432
SVM	0.06689	0.08850	0.92230
5-NN	0.12886	0.21023	0.83046

(c) Experiment C - BL vs 40% Load

Figure 7.3: Average Classification Results Per Learner

Experiment A

This experiment determines whether our classification models can detect hard or soft impacts based on the vibration data from six sensor channels. Figure 7.3(a) shows the results of this experiment. We can quickly see from this table that both decision tree classifiers were able to distinguish both percussion states (hard hammer HH and soft hammer SH) with a satisfactory degree of accuracy. We also observed that by tuning the parameters of the decision tree (C4.5N), we were able to gain a slight performance

increase over the decision tree model built using default parameters (C4.5O). The C4.5N model had a FPR of 0.00174 on HH and 0.00254 on SH meaning that on average, there were 17 false alarms (or, false positives) and 9,983 correctly identified faulty instances. This is significantly better than the 5-NN model which produced on average 1,190 false alarms and 1,884 missed or undetected faulty instances. In the live system, this could be equivalent to repeated impact from a submerged object or a marine animal where roughly 19% of the impacts go undetected; this can obviously be catastrophic.

Experiment B

The results for Experiment B are shown in Figure 7.3 (b). This experiment involved three baseline signals acquired at different speeds (52.5, 64 and 73 RPM). At each speed, we also acquired data for one faulty state (soft hammer SH). Here, the goal is to determine if abnormal states can still be detected with a high degree of confidence regardless of the rotational velocity of the dynamometer and by extension, the turbine. We find again that the two decision tree models yielded acceptable results. The 5NN model again had the worst performance.

Experiment C

For the third experiment, Experiment C, Figure 7.3 (c) shows similar results as the other two experiments. The MLP classifier, which performed third best overall for all three experiments is seen here to yield acceptable false negative rates and accuracy values. As before, both decision tree models were able to detect the addition of a counter-torque or load to the dynamometer with a high accuracy.

The explanation for the poor performance of the 5NN in all three experiments may lie in the way the algorithm handles missing values in WEKA. 81.92 % of the instances

in each binary classification file containing 20,000 instances (10,000 instances from the normal class and 10,000 instances from the abnormal class) have at least one missing value, of which the older instances in the batch will have a higher degree of missing values due to the nature of the algorithm (refer to Section 4.2.1). The nearest neighbors implementation in WEKA works by assigning a maximum distance for attributes with missing values. If the instance to be classified has multiple missing values, there is a greater likelihood for it to be misclassified since the instances that best match may be assigned greater distances due to the degree of missing values, and thus, never be selected as the nearest neighbor(s).

7.3.7 Data Imputation Results

To confirm that the behaviors of these learners are due to the missing values in the data, we present the results after imputing these missing values in the data from all three experiments using MI and EMI. These results are shown in Figures 7.4, 7.5 and 7.6.

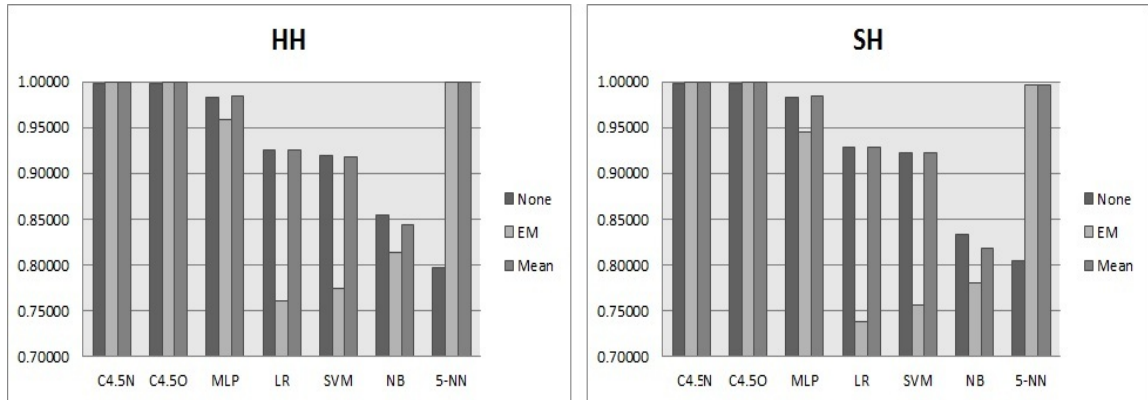


Figure 7.4: Average accuracy by learner before and after imputing data for Experiment A - BL vs HH and BL vs SH

The three bars in each cluster of each graph represent the accuracy values obtained from the original dataset (labeled None), Expectation-Maximation Imputation (EMI)

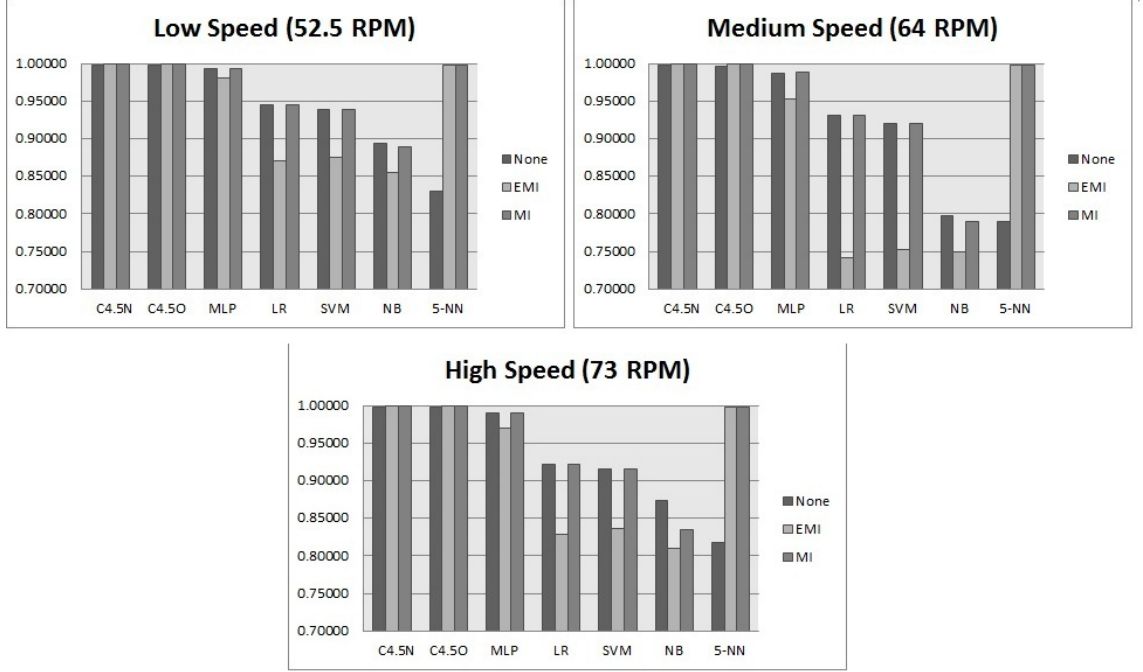


Figure 7.5: Average accuracy by learner before and after imputing data for Experiment B - BL vs SH where BL and SH recorded 3 times at different speeds

and Mean Imputation (MI) respectively. The first two graphs (Figures 7.4 and 7.5) show results for the HH and SH experiments (Experiment A). Figure 7.5 presents the results for Experiment B, with the first graph being the lowest speed, the second graph being the medium speed and the last graph the highest speed. The bottommost figure presents the results for Experiment C.

From the results, we find that the MLP, LR, SVM and NB models built on EMI imputed data performed worst while those built on MI imputed data were almost as good as those built on the original unimputed data. This is contrary to results found in previous studies comparing MI and EMI on other datasets [83] and is counterintuitive considering the relative lack of sophistication of the MI algorithm. For the C4.5N (decision tree with tuned parameters), C4.5O (decision tree with default WEKA parameters) and 5NN, both imputation techniques seemed to perform equally well,

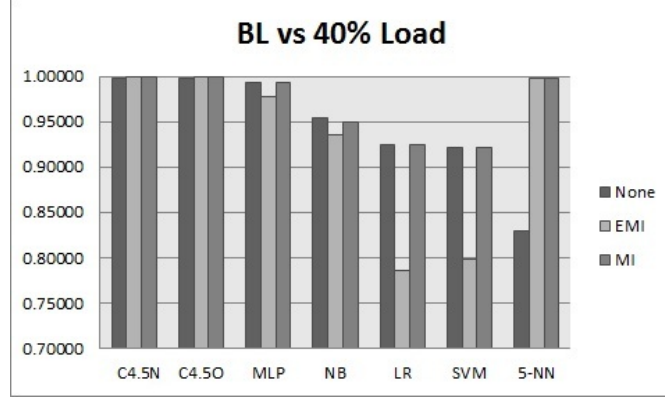


Figure 7.6: Average accuracy by learner before and after imputing data for Experiment C - BL vs 40% Load

with 5NN models greatly benefiting from the lack of missing values. Overall, the MI technique seems to be the better of the two imputation techniques for these datasets.

7.4 CASE STUDY 2 – FEATURE SELECTION

This section concerns feature selection on wavelet transformed vibration data. Wavelet transforms, which are becoming increasingly popular as a vibration analysis technique, are applied in this case study on vibration signals gathered from a dynamometer to convert the time series of raw vibration readings into a time series of resonance vectors. This dynamometer is the test bed for an ocean turbine prototype, which demands an automated monitoring solution such as an MCM/PHM system due to its complexity, unpredictable environment and other factors.

Each component of the data vector produced by the wavelet transform is coded as a feature. Feature-level data fusion [40] is applied to the wavelet output to combine the data from all vibration sensors into a single file for each experiment. This gives the advantage of having a unified view of the data and possibly improved classification performance.

In this section, we empirically investigate several feature selection techniques and

gauge their effects on the performances of some widely used classifiers on experimental data gathered from the dynamometer. From the results of our case studies, we hope to determine which feature selection technique(s) best identify the optimal subset of features that will enable a classifier to distinguish abnormal or faulty states from non-faulty states either as good as or even better than it would on the complete dataset. We also experiment with different sizes of the feature subsets to determine the optimal number of features to be selected. The results with and without feature selection are presented. During our experiments, we noted that the speed of some classifiers suffered due to the size of the entire dataset; this also inspired the need for feature selection.

7.4.1 Feature Selection on SWT Data

Thirty-two seconds of data were sampled from the dynamometer for this experiment the same way as in Experiment A of the previous section. As with that experiment, there were three scenarios baseline (BL), soft hammer (SH) and hard hammer (HH). As before, the Streaming Haar Wavelet Transform (SWT) was applied to that data to convert the time series of amplitude readings (in millivolts) into a time series of resonance vectors. The output of the Haar wavelet transform are 18 files (6 channels x 3 scenarios) each having 80,000 rows. In each row, there are 14 wavelet features corresponding to the frequency amplitude at different times and frequency resolutions. As previously noted, due to certain optimizations made for operating on streaming data, many instances have missing values.

In this study, feature level fusion is done by performing a union of the features across all six sensor channels for each setup/experiment, i.e. BL, SH and HH. The 14 wavelet features from channel 1 were combined with the 14 wavelet features from channel 2 for each setup and so on, producing 3 new files, each containing 84 features.

The wavelet features for each channel and the BL scenario were combined with those for each of the faulty classes, i.e. SH or HH. More specifically, the 80,000 instances of BL were appended to the end of the SH data file, and then these same 80,000 instances for BL were appended to the data file for HH. After doing this, we have 2 files = $\{SH, HH\} \times \{BL\}$. By combining the files in this manner, we prepare our dataset for a binary classification problem, meaning that each classifier will only need to distinguish between two classes at a time per channel – the normal state (BL) and a single fault.

We used the WEKA data mining tool to train seven learners (namely, Naive Bayes (NB), multi-layer perceptron (MLP), 5-Nearest Neighbors (5NN), Support Vector Machine (SVM), Random Forest with 100 trees (RF100), Logistic Regression (LR) and Decision Tree with default parameters (C4.5O)) on each data file and evaluate their ability to correctly identify which of the 160,000 instances in each file is red (faulty) or green (normal). One run of five-fold cross validation with feature selection (as described in Section 4.8) is implemented and applied to each of the two datasets (combined file with BL vs. HH and combined file for BL vs. SH) while building each classification model. For each feature selection technique, we build classification models based on datasets constructed using only the top 3, 4, 5, 7 and 10 wavelet features chosen by that technique. We also built classification models from both files using all 84 features, i.e., no feature selection. The AUC is used as the performance measure.

7.4.2 Results

The classification results for each learner are shown in Figure 7.7. In the figure, there are seven charts, one for each machine learner – Naive Bayes (NB), multi-layer perceptron (MLP), 5-Nearest Neighbors (5NN) Support Vector Machine (SVM),

Random Forest with 100 trees (RF100), Logistic Regression (LR) and Decision Tree (C4.5O). Each graph contains ten clusters of bars. The first eight bars represent the results for the eight feature selection techniques while the last group (labeled “None”) shows that classifier’s performance when no feature selection is performed. Within each of the first eight clusters, there are five bars representing the number of features n used to build the model. In order of appearances, these values of n are 3, 4, 5, 7 and 10.

From the graphs, the difference in performances across the different combinations of feature selection technique and classifier is quite evident, especially in the cases of Naive Bayes, LR, MLP and SVM where Chi-squared and Information Gain yielded the worst results. All classifiers except 5NN built perfect models (i.e. AUC of 1) on the dataset containing all 84 features. This information is shown in the graphs as the right-most bar.

For ease of analysis, we list the number of features and the feature selection technique(s) which yielded the best performance per classifier on our dataset in Table 7.1. In this table, we include the AUC that was obtained by that classifier on the reduced set of features selected by the feature selection technique. We found that by selecting just the top 3 features using either the Information Gain (IG) or Chi-squared (CS) filters, 5NN, C4.5O and RF100 were able to achieve perfect results, which in the case of 5NN was better than its performance on the dataset containing all features. Note that these two techniques had the opposite effect on the NB, LR, MLP and SVM classifiers as previously observed.

7.4.3 Feature Selection on STWTB Data

The baseline signal for this study was acquired with 45% resistive load and SFTA (the dynamometer drive shaft) turning at 25 RPM. Data was also acquired with a

Learners	No. of Features	FS Technique	AUC
NB	10	PRC	0.999998
MLP	7	PRC	0.999998
5NN	3	IG/CS	1
SVM	10	KS	0.998354
RF100	3	IG/CS	1
LR	10	KS/Dev	0.999108
C4.5O	3	IG/CS	1

Table 7.1: Summary of Feature Selection Techniques Per Classifier

90% resistive load applied at the same RPM and then with 45 and 90 percent load at 50 RPM. Data gathered at 25 RPM will be used to generate the classification models used in this study; these models will then be tested against the data collected at 50 RPM. The change in load is what we aim to detect and the loads are therefore coded as classes. We test against a different RPM here to ensure the robustness of these techniques in varying operational conditions (such as speed) since typically the speed of the turbine will vary during operation based on the flow velocity of the ocean currents.

Data from the accelerometers were sampled at 5,000 Hz for 4 seconds for each experiment, producing a total of 20,000 readings per experiment. These accelerometers are denoted as channels 1 through 6 (CH1, CH2, . . . , CH6) throughout this case study. The raw data consisted of 24 files = 6 channels x 2 loads x 2 speeds. Following feature-level fusion, this became a single data file for each combination of load and speed (e.g., four in total).

For this experiment, four machine learning techniques (Naive Bayes, 5-Nearest

Neighbors, Decision Tree with tuned parameters – C4.5N – and Logistic Regression) were trained to detect the underlying patterns in the vibration signatures and to predict the state of the machine. Seven models were built for each machine learner (or classifier) by training the classifier on the 78 features fused from 25 RPM data and then from just the top 2, 4, 6, 8, 10 and 15 features selected from that dataset by five feature selection techniques (AUC, Information Gain, Chi-squared, PRC and Signal-To-Noise). The classification accuracy (percentage of correctly labeled test instances) for all learners are presented in the next section.

7.4.4 Results

Table 7.4.4 shows the performances of models built during these experiments on the 25RPM data when evaluated on the 50RPM data. Each of the four rightmost columns represents the accuracies obtained by an individual learner. These learners are the 5-Nearest Neighbors (5NN), Decision Tree (C4.5N), Naive Bayes (NB) and Logistic Regression (LR). The first row of the table are the results when all features were used (no feature selection). The remainder of the table is divided into five sections where each section shows the results per feature selection (FS) technique and contains six rows; these rows present the results when 2, 4, 6, 8, 10 and 15 features were selected using that FS technique.

From the results obtained, we found that the AUC, Chi-squared (CS) and Information Gain (IG) techniques all produced the same feature rankings, and thus, the same performances for all four learners. The top 8 features selected by these techniques were sufficient to reproduce a perfect 5NN model. Of all the feature selection algorithms, these three yielded the best Decision Tree models using just 4 features.

It is of note that unlike any other learner, the Decision Tree performed worse built using a reduced feature set (even when IG is used to select the features) than when all

Table 7.2: Classification Accuracies per Feature Selection Technique, Learner and # of Features

Technique	# of Features	5NN	C4.5N	NB	LR
None	All	1	0.90493	0.891222	0.996573
AUC	2	0.967398	0.785809	0.914424	0.951207
	4	0.980728	0.868492	0.895781	0.977962
	6	0.998365	0.813097	0.502452	0.980854
	8	1	0.847554	0.50132	0.991889
	10	1	0.847554	0.801339	0.952779
	15	1	0.847554	0.805301	0.903986
CS	2	0.967398	0.785809	0.914424	0.951207
	4	0.980728	0.868492	0.895781	0.977962
	6	0.998365	0.813097	0.502452	0.980854
	8	1	0.847554	0.50132	0.991889
	10	1	0.847554	0.801339	0.952779
	15	1	0.847554	0.805301	0.903986
IG	2	0.967398	0.785809	0.914424	0.951207
	4	0.980728	0.868492	0.895781	0.977962
	6	0.998365	0.813097	0.502452	0.980854
	8	1	0.847554	0.50132	0.991889
	10	1	0.847554	0.801339	0.952779
	15	1	0.847554	0.805301	0.903986
PRC	2	0.5	0.4917	0.689009	0.5
	4	0.5	0.4917	0.688286	1
	6	0.5	0.4917	0.500252	1
	8	0.5	0.4917	0.993901	0.983558
	10	0.5	0.4917	0.984123	0.987739
	15	0.506791	0.4917	0.670649	0.999874
S2N	2	0.5	0.798447	0.571963	1
	4	0.5	0.4917	0.570202	1
	6	0.5	0.4917	0.563663	1
	8	0.608463	0.4917	0.890751	1
	10	0.964569	0.4917	0.88374	0.989751
	15	1	0.4917	0.84243	0.908325

78 features/attributes were used. This makes sense considering that the Decision Tree algorithm performs its own feature selection, so, when presented with all features, it can determine the optimal subset of features to construct a model. Also, the feature ranking produced by the IG ranking may be different from the feature ranking in the Decision Tree, and thus using IG as a feature ranker may yield different results than using the Decision Tree to select its best features from the entire feature set. IG as a feature ranking technique considers each attribute's individual contribution to the entire dataset while the IG-based feature selection step that is inherent to the Decision Tree construction process ranks attributes based on how they perform together. Given m instances in the training set, the IG feature ranker evaluates each attribute a based on all m instances while the Decision Tree evaluates that attribute on the subset of instances that fall within the current branch of the tree.

The NB learner gave near perfect results when trained on a subset of the features selected using the PRC technique with 8 features being the optimal number of features. The LR learner built a perfect model from just 2 features selected by the S2N technique. The models these two learners built on the datasets using these reduced feature spaces yielded better results than those built using all features. The NB model was as much as 10% better on the dataset containing just 8 features than it was on the dataset with all features. The false positive rate (FPR), or the percentage of false alarms, decreased from 21.46% to just 0.26%. In terms of an ocean turbine, false alarms cause unnecessary system shut down and maintenance trips (resulting in a loss of time, money and uptime).

Overall, 3 out of the 4 learners in this study stand to benefit from feature selection; the fourth learner (the Decision Tree) suffered no major loss in performance. While the number of features are small in this study, note that this number differs according to the window size of the wavelet transform. In future studies, we will experiment

with different window sizes and how these affect our results. Also, although we did not bench test the evaluation speeds of each of our models, we realized better evaluation times for the 5NN model when using the reduced set of features than with all features. Considering the quantity of data that would be analyzed per sensor within a minute ($5,000 \text{ Hz} \times 60 \text{ seconds} = 300,000 \text{ data points}$) on a deployed ocean turbine, these time savings would free up computing resources for other processes involved in the MCM/PHM system such as prognostics and system advisory generation.

7.5 CONCLUSION

This chapter presented three case studies which investigate the ability of different machine learning techniques to build reliable classification models on experimental data gathered from a dynamometer. This dynamometer is a test bed for the drivetrain of an ocean turbine prototype, and experiments conducted on data gathered from it reveals invaluable insight needed for developing a data-driven MCM/PHM system for the turbine.

From the three experiments in the first case study, which revolved around analyzing learner behaviour on data with missing values, we found that the decision tree learner was able to consistently identify abnormal states with the greatest degree of accuracy and confidence. An interesting note is that on the unimputed data, the nearest neighbor model performed the worst of the seven classifiers on all experiments. This learner was shown in [72] to have an overall good performance, and even outperformed decision tree models in some of those studies. This very point stresses the importance of experimenting with multiple machine learning techniques to determine the optimal approach for different applications and datasets.

The last two studies compared the performances of several machine learners when

trained on all features against their performances when trained on a subset of n features selected by different feature selection techniques. Features were extracted using the Streaming Wavelet Transform (SWT) for the first of these two studies and via the Short Time Wavelet Transform with Baselineing (STWTB) for the second. From the results, it is clear that learners respond differently to some feature selection algorithm than they do others. Findings from the SWT feature selection case study showed that the 5-Nearest Neighbors, Random Forest and Decision Tree algorithms were able to build perfect models using just the 3 most important features (3.33% of the total number of features) selected by either the Information Gain or Chi-squared techniques. On the more complex study using a test set acquired at a different rotational velocity than the training set, the Decision Tree no longer fares as well as it did in the previous study but 5-Nearest Neighbors still yielded perfect models using the top 8 features (10.25% of the feature set) selected by the same two algorithms.

In the next chapter, we will employ this STWTB implementation again for preprocessing the raw vibration waveforms. This STWTB approach was found to produce robust models even when the current environmental or operating conditions differ from what they were when the training data was first acquired. The next chapter considers another type of data imperfection in form of class imbalance, a known issue in condition monitoring applications.

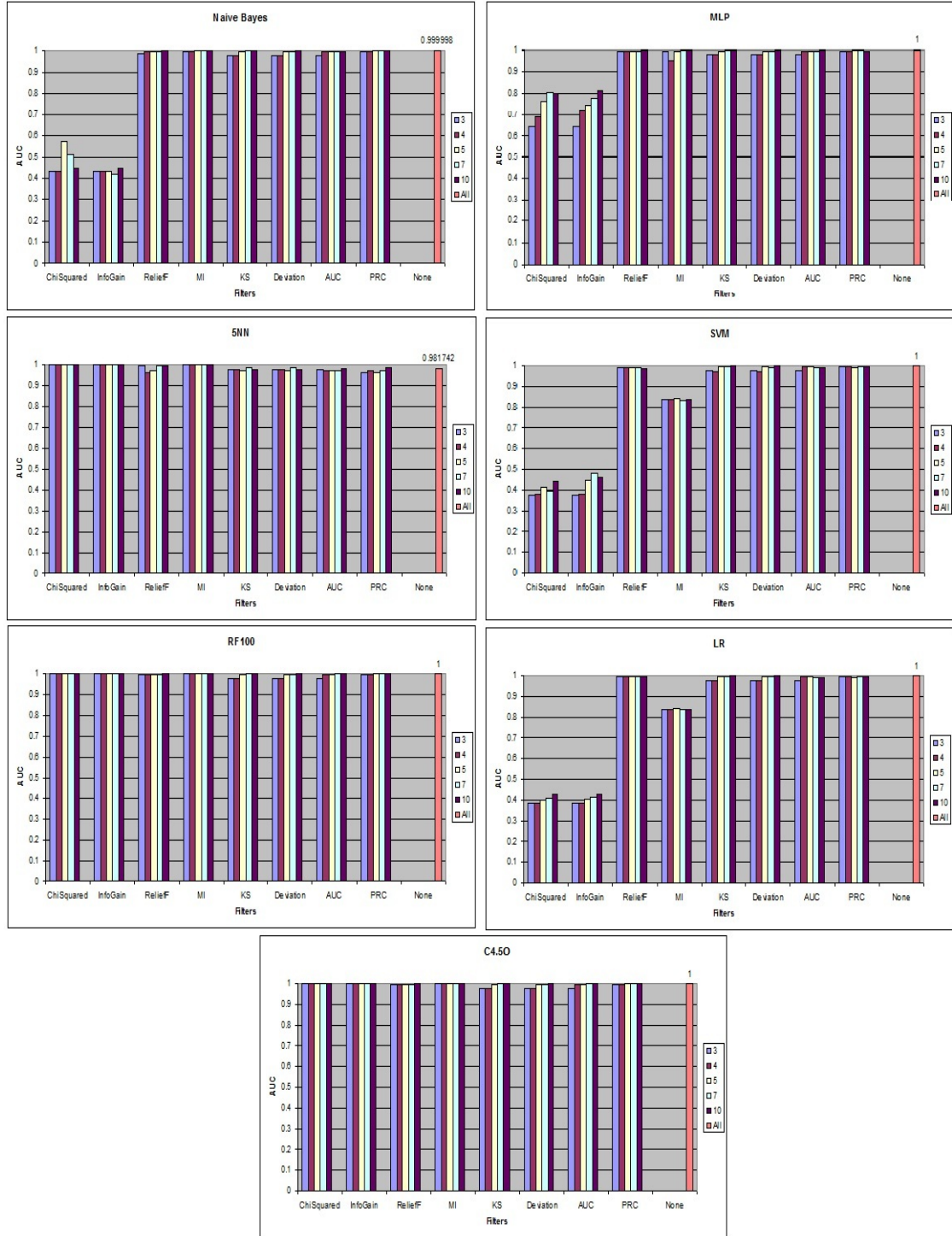


Figure 7.7: AUC per Learner and Feature Selection Technique

MCM/PHM In The Presence of Class Imbalance

8.1 INTRODUCTION

Because of the expenses associated with developing the ocean turbine and its dynamometer (designed to test components of the turbine), there are constraints placed on the types of system tests that can be conducted. All tests must be unobtrusive and cannot in any way damage the machines. This means that certain types of fault scenarios, such as a bent shaft, bad bearing assemblies or an imbalanced rotor, may never be modeled on the dynamometer or the turbine itself; it may also take weeks, months or even years for the turbine to develop certain types of faults or problems on its own. Once a fault does develop in the turbine, remediation steps will be taken (by an operator or by the turbine's MCM/PHM system) as soon as the problem is detected to avoid damage to the expensive machine. Thus, only a small amount of data can be collected while the turbine is in this faulty state. Consequently, there is a lack of “faulty” data corresponding to many of the possible abnormal states and researchers are left with a minimal allowable set of test cases and a negligible amount of faulty data. Considering the large quantity of data being collected on a regular basis, especially from vibration sensors which produce thousands of readings per second, the ratio of abnormal to normal occurrences can be as small as 1:1000 (or 0.01%) [150].

This difference in the quantity of normal versus abnormal data has been shown in many other machine learning applications to result in the creation of suboptimal classification models which label most, if not all, of the data as belonging to the majority class (*i.e.*, normal). This suboptimal ratio of classes is called *class imbalance* and is found in many real-world datasets. Class imbalance occurs when there are significantly fewer instances in one or more classes in a dataset compared to the number of instances in the remaining classes. When trained on imbalanced datasets, traditional machine learning techniques tend to simply ignore the minority class(es) and label all instances as being of the majority class to maximize accuracy.

This problem has been studied in many domains [72, 75] but there is little or no research related to the effect and remediation of class imbalance in fault data for condition monitoring of an ocean turbine. An investigation presented in this paper demonstrates how this class imbalance problem could affect reliable state detection within a data driven MCM/PHM system for an ocean turbine. The study also shows how various degrees of imbalance affect the abilities of seven machine learners to distinguish between two states of an ocean turbine. To do so, we empirically evaluate the performances of seven widely used, but very different, machine learning algorithms when trained on datasets with varying class distributions to distinguish between a normal and an abnormal state.

All data used in these experiments were collected from the testbed for an ocean turbine and were under-sampled to simulate the different levels of imbalance. The raw vibration signal acquired from each sensor was pre-processed using the Short Time Wavelet Transform with Baselineing (STWTB) algorithm (described in Section 4.2.1). From the previous chapter, it was found that STWTB transformed data allowed learners to build models that were robust enough to reliably identify changes in system state even if the current rotational velocity is different from what it was

when the training data (used to generate the model) was collected. Here, we assess whether models generated from STWTB data are tolerant of class imbalance.

For the specific “abnormal” conditions that are considered in the experiments described later in this chapter, it was possible to collect the same amount of data in those abnormal state as was collected in the normal state (meaning the class distribution of the dataset was originally balanced), since these abnormal states are unobtrusive and will not actually damage the test equipment (i.e. the dynamometer). This will not be true of naturally occurring abnormal or faulty scenarios, which could result in system failure. Hence, we introduce artificial class imbalance into our dataset to simulate the problem of having such imbalanced data. Two sensor fusion techniques were applied in these case studies to combine the readings from the six sensor sources. An analysis of how each learner performed when trained on data fused with either of the two techniques and how the class distribution affected these results was also conducted.

8.2 CONTRIBUTIONS

The main contributions of this chapter are:

- The case studies in which experimental data are used to answer key research questions related to class imbalance in and fusion of ocean turbine sensor data
- A first assessment of the robustness of models generated using data pre-processed by this wavelet-based baseline-differencing technique when the class distribution is imbalanced.
- An empirical comparison of data fusion approaches for dealing with class imbalance

- The number of machine learners evaluated in these experiments. Other studies related to class imbalance and data fusion involved three or fewer learners [142].
- To the authors' knowledge, our research is the first studying the class imbalance problem in this domain.

The remainder of this chapter is organized as follows. Section 8.3 presents a background into the class imbalance problem. Approaches to handling the class imbalance problem in other reliability analysis applications are surveyed in Section 8.4. Two case studies shown in Section 8.6 demonstrate how class imbalance affects classifier performances for distinguishing between two states when applied to experimental data collected from an ocean turbine's dynamometer with and without data fusion. Results are discussed in Section 8.7 and concluding remarks are given in Section 8.8.

8.3 BACKGROUND

The occurrence of imbalanced or skewed datasets is a common challenge in data mining applications. Such datasets exist in both binary (two class) and multi-class (more than two classes) classification problems where the goal is typically to detect one or more rare events, such as the presence of ailments, seismic activities, faults or failure states in machinery, network intrusions, etc. The disparity in the number of these rare cases versus the abundance of observations made under normal circumstances typifies an imbalanced dataset.

When presented with an imbalanced dataset, traditional machine learners trained to distinguish between the different classes tend to ignore the minority class and simply label new examples (instances) that actually are in that minority class as belonging to the majority class. This occurs because these algorithms strive to maximize overall accuracy (percentage of correctly classified instances). For instance, given a

training set containing 1,000 instances where 1% (10) of the instances are in class A and the remaining 990 are in class B, a learner can achieve a 99% accuracy by classifying all instances as being in class B since it would have correctly labeled 990/1000 instances. A model's accuracy, therefore, is not a good indicator of its performance on imbalanced data.

This *class imbalance* problem, its effect on data mining and machine learning techniques as well as approaches to learning from imbalanced datasets have been well studied in many domains, such as in software quality prediction [60], protein classification [165], wind turbine fault detection [150], IP traffic analysis [23] and text classification [97]. The vast amount of research effort invested into class imbalance across these and other domains has been summarized in numerous papers [62, 65], which also list techniques that were proposed and implemented for dealing with the class imbalance problem. Such techniques are usually classified according to whether they are applied at the data level or at the algorithmic level. Data level approaches aim to balance the class distribution while algorithmic approaches involve adjusting the learning algorithm itself to take into account the natural class distribution.

8.3.1 Data level approaches

Data level or re-sampling techniques are the most widely used approaches to handling the class imbalance problem because they are simple to apply. Approaches that fall in this category work by changing the overall class distribution via either over-sampling, under-sampling or a combination of both.

Over-sampling of the minority class involves the introduction of new instances to that class, usually by replicating existing examples or creating new instances. In random over-sampling (ROS), examples from the minority are selected at random and duplicated until the class distribution is at the desired level. Other over-sampling

approaches, such as Synthetic Minority Over-sampling Technique (SMOTE) [15], aim to increase the number of minority instances in a more intelligent manner: by using knowledge of the characteristics of the data itself to create new data points.

Under-sampling approaches typically involve the removal of instances from the majority class. As with over-sampling, this can be done randomly (known as random under-sampling or RUS) or intelligently. Data cleaning techniques such as one-sided selection [72], Tomek links [143] and neighborhood preprocessing [16] are a subset of intelligent sub-sampling approaches; these aim to remove redundant and noisy data from the majority class.

Both over-sampling and under-sampling have their pros and cons. Over-sampling tends to lead to over-fitting, where a classification model is too specific to the training set and fails to generalize to and correctly label new data. Also, since there is now more data to process, the time taken to train classifiers on the over-sampled dataset will also increase. On the other hand, because there is less data to process, under-sampling reduces the time needed to build classification models. However, when removing instances using under-sampling, useful data may also be discarded resulting in a loss of information.

8.3.2 Algorithmic level approaches

Algorithmic level solutions include adjusting the classifier’s decision threshold, tuning the machine learner parameters, assigning costs to the different classes (cost sensitive learning) and applying a one-class learning scheme. The classifier’s decision threshold is a parameter it uses when labeling a new instance. For a binary classification problem, if the posterior probability of the class of interest exceeds this threshold then the instance is labeled as being in the minority class; otherwise, the instance is labeled as being in the majority class. By lowering this threshold, a classifier will be

more likely to label instances as the minority class.

Cost sensitive learning [93] involves assigning a cost to each class to account for the relative importance of the classes in relation to the problem so that there is a greater cost for incorrectly labeling (or misclassifying) a minority instance. Cancer gone undetected, for example, is significantly worse than a false alarm as the patient may die due to lack of treatment. Error costs are, however, difficult to assess and are usually unknown.

In one-class or recognition based learning, a classifier is trained to identify a single class of interest instead of distinguishing between two or more states (discriminative learning). When it encounters a new instance, the classifier determines whether the instance belongs to the class of interest or not. Because the classifier only considers a single class, typically the minority class, the abundance of instances in the majority class should not affect its performance.

8.4 RELATED WORK IN RELIABILITY ANALYSIS APPLICATIONS

While previous surveys discuss the class imbalance problem in general, the focus here is to briefly review studies involving skewed datasets in the field of reliability analysis, specifically in fault detection and condition monitoring, to reveal the state of the art in dealing with class imbalance in this domain.

Zou *et al.* utilized a cost-sensitive approach for training a Support Vector Machine (SVM) for steam turbine fault diagnosis [163]. For their multi-class problem, the researchers combine several SVM learners – originally designed for binary classification – using a cost conscious One Versus All (OVA) and One Versus One (OVO) method for fusing the decisions of the generated classification models. Given k classes, the OVA method involves generating k classifiers on data from two classes where the i^{th}

SVM is trained to label the i^{th} class as positive and all others as negative. Using OVO, $k(k-1)/2$ SVM models are constructed and the overall decision is made using a voting strategy. They also introduce a reject option to ignore classified instances where the computed reliability values are below a specified threshold.

Verma and Kusiak [150] applied a Tomek links-based data sampling approach to remove borderline and noisy data from the majority class for fault detection in wind turbines. This study considers two classes: a normal and a faulty state (with the faulty instances accounting for 10% of the training set). In their approach, a majority instance is removed from the training set (the dataset used to build the classification models) if there exists a minority instance such that the Euclidean distance between the two instances is less than the smallest Euclidean distance between this minority data point and any other instance in the minority class. The Tomek links under-sampled data is then passed through a Random Forest data sampling algorithm to further reduce the degree of class imbalance. This Random Forest sampling algorithm also discards noisy, borderline and redundant samples from the majority class based on any instances that are incorrectly labeled by Random Forest models generated by removing random subsets of the majority dataset.

In [94], Liu and colleagues proposed a modified SVM learning algorithm for multi-class classification of blast furnace fault data. The four faulty states accounted for 3.5%, 4.8%, 6.7% and 7.2% of the training set while the remaining 77.8% of the instances were normal. The proposed algorithm is a multi-step iterative process. As in the Verma-Kusiak study, instances are pruned from the majority class based on their distance from instances in the minority class. In the next step, unlabelled instances are selected and added to the training set to compensate for the removal of instances in the previous step. Lastly, noisy instances are weeded out using the Edited Nearest Neighbor algorithm [119].

8.5 METHODOLOGY

The training set is the set of labeled observations that are used to initially train or build the classification models. In other words, the actual class to which each observation belongs is already known. The test set consists of new observations that the models are applied to. Each observation is then labeled by the model as being representative of one or more states (classes) from the set of possible classes C . Although in the live MCM/PHM system, the actual class of new sensor observations will be unknown, the class labels of all instances in the test set is maintained for the purpose of evaluating these models.

8.5.1 Learners

Seven machine learning algorithms are evaluated on their ability to distinguish between two operating states of an ocean turbine dynamometer from sensor data acquired from the machine. Classification models were constructed in WEKA, an open source data mining software package. The seven algorithms, described in greater detail in Chapter 4, are listed below.

- **Naive Bayes learner** (NB) – applies Bayes’ rule of conditional probability and a “naive” assumption of independence among the features to predict the probability that an instance belongs to a specific class.
- **Decision Tree** (C4.5) – a tree-like machine learning model. We built this classifier using default values.
- **Logistic Regression** (LR) – labels an instance in a binary classification problem based on the measured probability of the class of interest (which in our case is the faulty scenario), similar to Naive Bayes.

- **k -Nearest Neighbors** (5-NN) – this lazy learner classifies a new instance by taking a distance-weighted majority vote of the classes of the k instances (for these experiments, we used $k = 5$, hence our nomenclature) in the training dataset that are closest to the new instance within the feature space.
- **Random Forest** (RF) – an ensemble learner composed of multiple unpruned decision trees. The default parameter values for the WEKA implementation of the Random Forest learner were used to construct this classifier.
- **Multi-Layer Perceptron** (MLP) – a form of feed-forward neural network.
- **Support Vector Machine** (SVM) – a hyperplane which divides a set of instances into two classes with maximum margin.

8.5.2 Feature Level Fusion

Our case studies investigate feature level and decision level fusion of the wavelet transformed vibration data. To fuse the sensor data at the feature level, a set union of the features produced by the wavelet transform from all channels was performed, which, intuitively, should improve a classifier’s ability to perform state detection since all the available data is being taken into account during the data mining process.

8.5.3 Decision Level Fusion

In these studies, the decision-level fusion process (Section 4.3.3) is repeated using each of the seven previously described machine learning techniques to see which generates the best classification models.

8.6 EXPERIMENTAL SETUP

The case studies presented in this section investigate the effect of class imbalance on different machine learners for performing state detection based on vibration data. Vibration readings of rotating machinery contain distinct signatures which can be used to determine the state of a machine. By comparing acquired signals against a known baseline signal, we can determine whether the machine is operating in an abnormal state. The data used in these experiments were all acquired from the test platform for an ocean turbine – a 20 kW onshore dynamometer designed for testing the turbine’s generator and drive train. In the case studies described below, we show how classification performances differ when attempting to detect when the resistive load being applied to the dynamometer has increased from 45% to 90%, where 45% load is considered to be normal. This is similar to having twice the normal load applied to the ocean turbine, as could occur if the on-board electrolysis device or other such devices meant to consume or utilize the power being generated by the turbine is unexpectedly drawing more energy.

8.6.1 Data Acquisition

Data collection and time synchronous averaging [90] of the vibration signals is performed by a WaveBook Data Acquisition Unit. The baseline signal for this experiment was acquired with 45% resistive load and SFTA (the dynamometer drive shaft) turning at 25 RPM. Data was also acquired with a 90% resistive load applied at the same RPM and then with 45% and 90% load at 50 RPM. In the first case study, data gathered at 25RPM will be used to generate the classification models, which were then tested against the data collected at 50RPM. In the second study, the 50RPM data is used as the training set while the 25RPM data serves as the test set. Again, we stress

that the change in load is what we aim to detect. We test against a different RPM here to ensure the robustness of these techniques in varying operational conditions (such as speed) since typically the speed of the turbine will vary during operation based on the flow velocity of the ocean currents.

Data from the accelerometers were sampled at 5,000 Hz for 4 seconds for each of the four setups (i.e., configurations of load and RPM), producing a total of 20,000 readings per setup and sensor. This means that, initially, the number of instances representing each class (i.e., load) is the same and the class distribution is balanced. The accelerometers are denoted as channels 1 through 6 (CH1, CH2, ..., CH6) throughout this case study. The resulting dataset consisted of 24 files = 6 channels \times 2 loads \times 2 speeds.

8.6.2 Data Transformation

A Short Time Wavelet Transform with Baselineing (STWTB) technique described in Section 8.6.2 was applied to each file separately. In the baselineing step in the STWTB process, a baseline is generated from the 45% load data for each source and RPM, and then subtracted from all the data (data collected at both 45% and 90% loads) observed by that source at the same RPM. This is done twice, once for each RPM. This baselineing step was deemed necessary to remove those portions of the vibration signals that are characteristic of the speed of the system, so that the remaining signal only depicts the vibrations caused by actual abnormalities in the machine (and not its operating conditions). For each file processed, this algorithm outputs 15,904 instances and 13 numeric features, with lower values for a specific frequency and instance indicating a close similarity to the baseline signal.

Class Distribution	% Positive Instances	% Negative Instances	Total # of Instances	# of Positive Instances	# of Negative Instances
A	0.1	99.9	15000	15	14985
B	0.5	99.5	15000	75	14925
C	1	99.0	15000	150	14850
D	2	98.0	15000	300	14700
E	5	95.0	15000	750	14250
F	50	50.0	15000	7500	7500

Table 8.1: Training Set Distributions

8.6.3 Class Distributions

The primary intention is to produce a state detection system capable of distinguishing between two operating states (45% load and 90% load) regardless of the turbine's RPM (or environmental condition). This is a simplified version of the real problem in which there will be multiple possible system states. Multi-class classification, or the problem involving detecting more than two system states, will be considered in future studies. To prepare the datasets from each experiment for binary classification (determining which of two states a particular set of observations belong to), the data gathered at 45% load is appended to the corresponding file containing the data gathered at 90% load for the same channel and RPM. This means there are now 12 files per experiment – 6 files (one for each channel) per RPM.

Random under-sampling of both classes in the 25RPM and 50RPM files was done to simulate the varying class distributions, meaning that instances were selected at random and discarded from the datasets gathered at both speeds. **Two** case studies were conducted: one with the 25RPM data being the training set and the 50RPM data used as the test set and the second with the 50RPM data used for training and the 25RPM data used for testing. Six different class distributions were simulated in both studies, as shown in Table 8.1. For each case study and class distribution, the

random under-sampling process was repeated 10 times to ensure that the observed results were unrelated to the exact subset of instances selected. The total number of instances was selected arbitrarily to be 15,000.

With the highest degree of skew, class distribution A is quite similar to what one would typically see in a fault detection application, and is most characteristic of the real world data than the other distributions. This is true especially in the case of vibration data, where data is sampled at a high frequency, usually more than 1,000 readings per second. Thus, it is important to understand the effect of such a highly skewed class distribution on techniques that would potentially be used in a condition monitoring system for an ocean turbine.

Seven machine learners – 5-Nearest Neighbors, Decision Tree, Logistic Regression, Multi-Layer Perceptron, Naive Bayes, Random Forest and Support Vector Machine – were trained on the 25RPM data (Case Study 1) and on the 50RPM data (Case Study 2) to distinguish between the two states. Models were evaluated against the other RPM for each case study; models built on the 25RPM data were evaluated against the 50RPM data and vice versa. There were a total of 360 models per learner in each study = 10 repetitions \times 6 channels \times 6 class distributions.

8.6.4 Performance Measure

The performances of the classifiers on these datasets are expressed in terms of the False Positive Rate ($FPR = \frac{FP}{N}$), the False Negative Rate ($FNR = \frac{FN}{P}$) and the area under the ROC curve (AUC). These performance measures are defined in Section 4.7.

As previously defined, the decision threshold is used by classifiers when assigning a class label to a new instance; if the confidence in labeling an instance as being faulty or abnormal exceeds this threshold then the instance is labeled as faulty. Lowering this threshold increases the TP and decreases the TN, making the classifier more biased

to the faulty (or positive) class. Varying the threshold, therefore, affects the FPR and FNR and the optimal value of this parameter depends on the class distribution and the data itself. Since there will be no single threshold that will work best for all class distributions, the best threshold for each of the 720 models built by each learner for both studies is determined by iteratively evaluating each model as the value of that parameter is gradually incremented from 0 to 1. The FPR and FNR presented in the results section are the best values found by varying the threshold for each model. The AUC, however, does not depend on a decision threshold since the ROC curve considers every possible decision threshold. This makes the AUC an excellent performance measure especially in class imbalance applications.

8.7 RESULTS & ANALYSIS

The discussion in this section answers the following research questions:

1. How does class imbalance in these datasets affect the results obtained when using the specified waveform analysis and machine learning techniques for performing fault detection on ocean turbine data?
2. Which learners are affected most and least by class imbalance in these datasets?
3. Which data fusion approach is better suited for this problem?
4. Which learners perform best regardless of the speed at which the training data was collected?
5. What is the optimal combination of data fusion type (feature level or decision level) and learner that will allow for reliable state detection regardless of class distribution and environmental changes (in this case, the dynamometer speed)?

Figures 8.1 and 8.2 shows the performances of the seven learners when trained on the datasets with varying class distributions. The three bar graphs in Figure 8.1 show the results when trained on 25RPM data while the three graphs in Figure 8.2 show the results when trained on 50RPM data. The two topmost graphs present the average AUC for each learner when no fusion technique is employed. Feature level fusion results and decision level fusion results are shown in the middle and bottommost pair of graphs respectively.

All graphs are laid out similarly and contain seven clusters, one for each learner, of six bars (representing the class distributions arranged from lowest to highest degree of imbalance). The first bar in each cluster is the average AUC obtained by the model when trained on the balanced dataset (distribution level F in Table 8.1).

8.7.1 How does class imbalance in these datasets affect the results using the specified waveform analysis and machine learning techniques?

From the results presented in Figures 8.1 and 8.2, we find our 5-NN models producing perfect results on the balanced feature-level fused data – middle graph in the left column – which is consistent with findings from a previous study conducted by our group [47]. These 5-NN models are unaffected by the class distribution. The SVM results on the balanced unfused data, though, are considerably different than were previously observed. This may imply that the performance of the SVM depends on the amount of available training data. On the feature-level fused data with an equal number of positive and negative instances, the SVM now yields perfect models for both training sets. The SVM models built on the 50RPM data appear unaffected by the class distribution; on the 25RPM data, the SVM models built using class distribution E (5% positive instances) perform worse than those built using the balanced data, especially on the feature-level fused data. The SVM’s performance on the

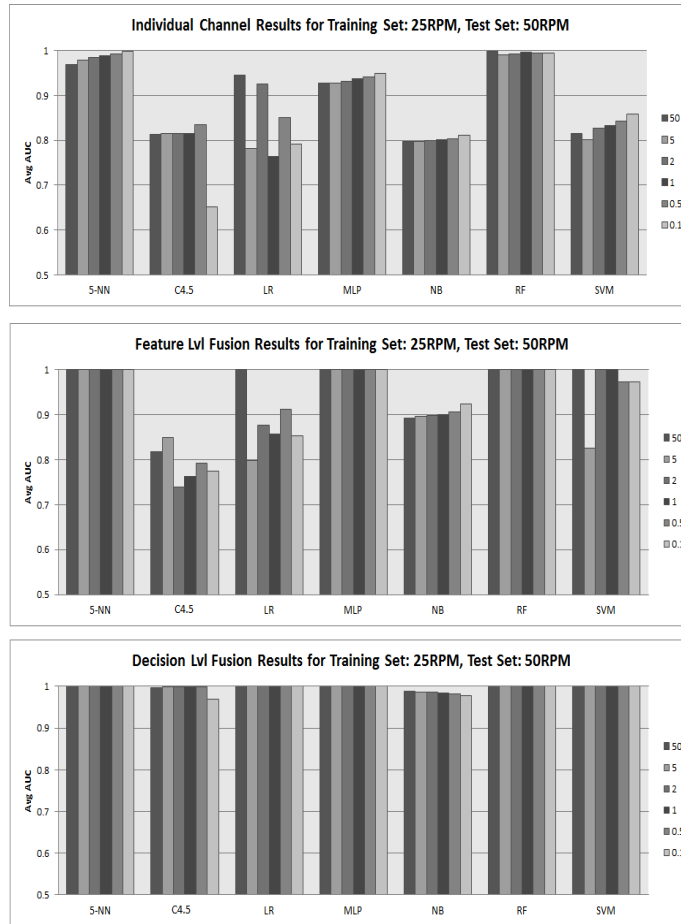


Figure 8.1: Classification Results (AUC) For All Learners With 25RPM Training Set

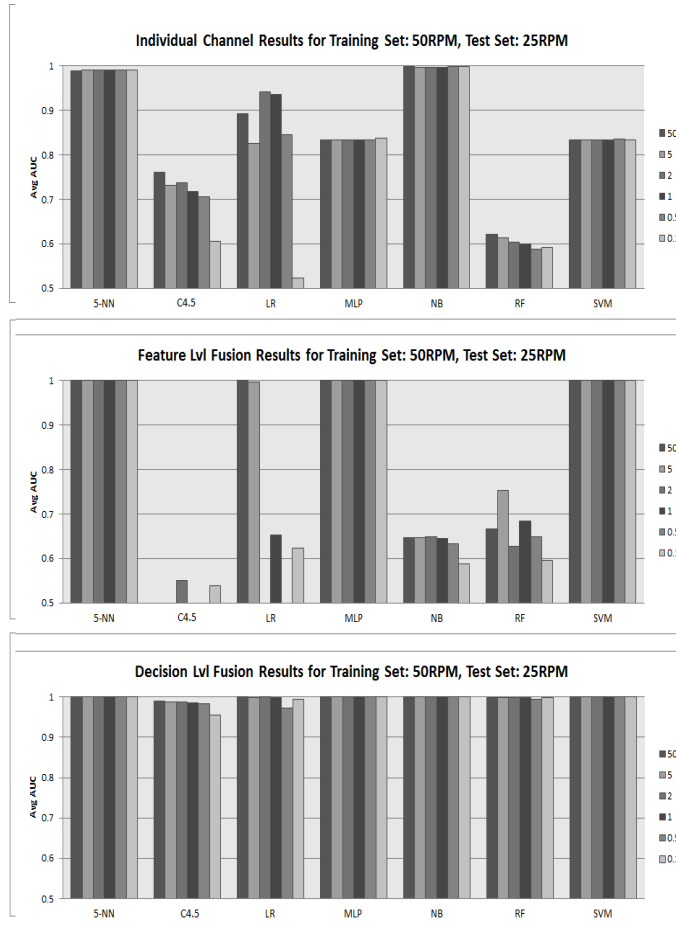


Figure 8.2: Classification Results (AUC) For All Learners With 50RPM Training Set

25RPM training sets with less than or equal to 2% of the instances being abnormal (positive class) is similar to its performance on the balanced data.

8.7.2 Which learners are affected most and least by class imbalance in these datasets?

To best determine how the learners are affected by the class distributions, we consider their performances on the unfused data (topmost pair of graphs in Figures 8.1 and 8.2).

Of the seven learners, the decision tree (C4.5) and Logistic Regression (LR) seem to be least tolerant of the degree of class imbalance in the training sets. The C4.5 models performed similarly for the first five class distributions (distributions A to E from Table 8.1) for both training sets, but when trained on the datasets with 0.1% of the instances being positive (class distribution F), the efficiency of this algorithm is obviously affected. On the 25RPM training set, the average AUC value is 0.2 less on the skewed dataset than it is on the balanced dataset. This difference is similar on the 50RPM training set, where the average AUC on the balanced set is 0.15 more than that on the highly skewed dataset. LR models perform even worse on the highly skewed 50RPM training set, with the average AUC being barely more than 0.5. Such a low AUC indicates that LR models are performing just as good as a random guess on that dataset.

Naive Bayes (NB) seems to be the most consistent learner for the unfused data in both case studies. Although its performance on the 25RPM data is worse than most, the degree of class imbalance barely, if any at all, affects the NB models.

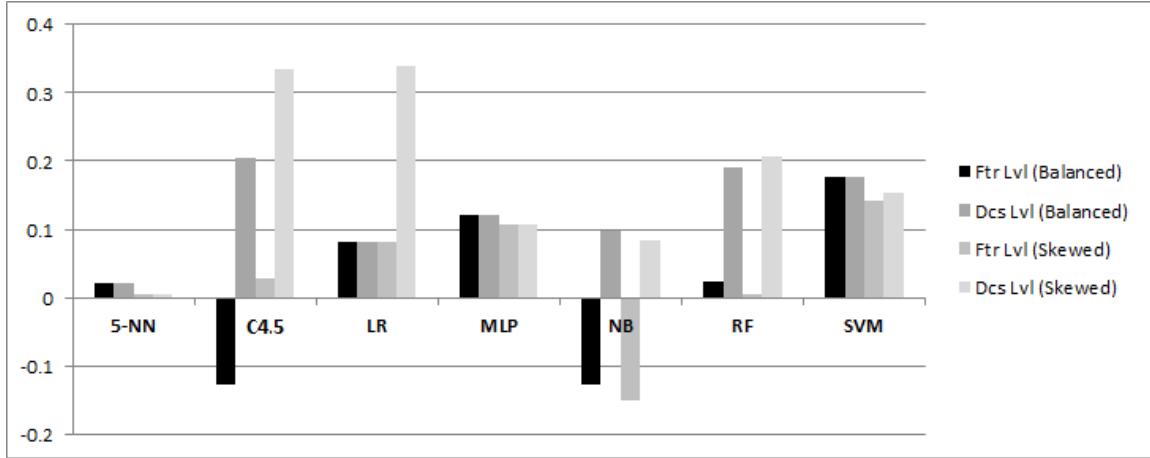


Figure 8.3: Change in AUC For All Learners On Balanced and Imbalanced Fused Data

8.7.3 Which data fusion approach is better suited for this problem?

First, we consider how the performances of the different learners are affected by each type of fusion. Figure 8.3 graphically shows the differences in the performances of models built on unfused data (in terms of the AUC) from those of the models built on the feature-level fused and decision-level fused data. This comparison is made for two data distributions: balanced (class distribution F) and highly skewed (class distribution A). This figure is also a clustered bar graph where the seven clusters are the various machine learners. The four bars in each cluster are:

1. Comparison on unfused versus feature-level fused balanced datasets averaged for both case studies
2. Comparison on unfused versus decision-level fused balanced datasets averaged for both case studies
3. Comparison on unfused versus feature-level fused datasets with the highest degree of skew (0.1% positive instances) averaged for both case studies

4. Comparison on unfused versus decision-level fused datasets with the highest degree of skew (0.1% positive instances) averaged for both case studies

If the models performed better on the data fused using the specific technique than they did on the unfused data, the bar for that corresponding relationship is above the y -axis; inversely, models which performed better on the unfused data are those where the bar is below the x -axis. The larger the performance difference, the higher or lower the bar will be.

From this graph, it can be seen that of all the learners, the decision tree algorithm showed the greatest improvement when trained on decision-level fused data (for both class distributions) and benefited most from this type of sensor fusion. The decision tree and Naive Bayes (NB) models performed better on average on the unfused sensor data than they did on the balanced feature-level fused data; this was also the case for the NB models when class imbalance was present. Both learners, however, performed best on the decision level fused data. Some learners, namely the Multi-Layer Perceptron (MLP), k -Nearest Neighbors (5-NN), Logistic Regression (LR) and Support Vector Machines (SVM), saw the same degree of improvement for both fusion techniques when the dataset was balanced. Overall, for both the balanced data (first two bars in each cluster in Figure 8.3) and the highly skewed data (last two bars in each cluster in the same figure), decision-level fusion performs either similar to or better than its feature-level counterpart for all learners in terms of the AUC.

In Figure 8.4, the class distribution is plotted against the average error rates (FPR and FNR) observed for each fusion technique (unfused, feature-level fusion and decision-level fusion) and training set (25RPM and 50RPM). These combinations are the six graphs in that figure. From this diagram, we find that decision-level fusion yields the lowest false positive and false negative rates and this observation

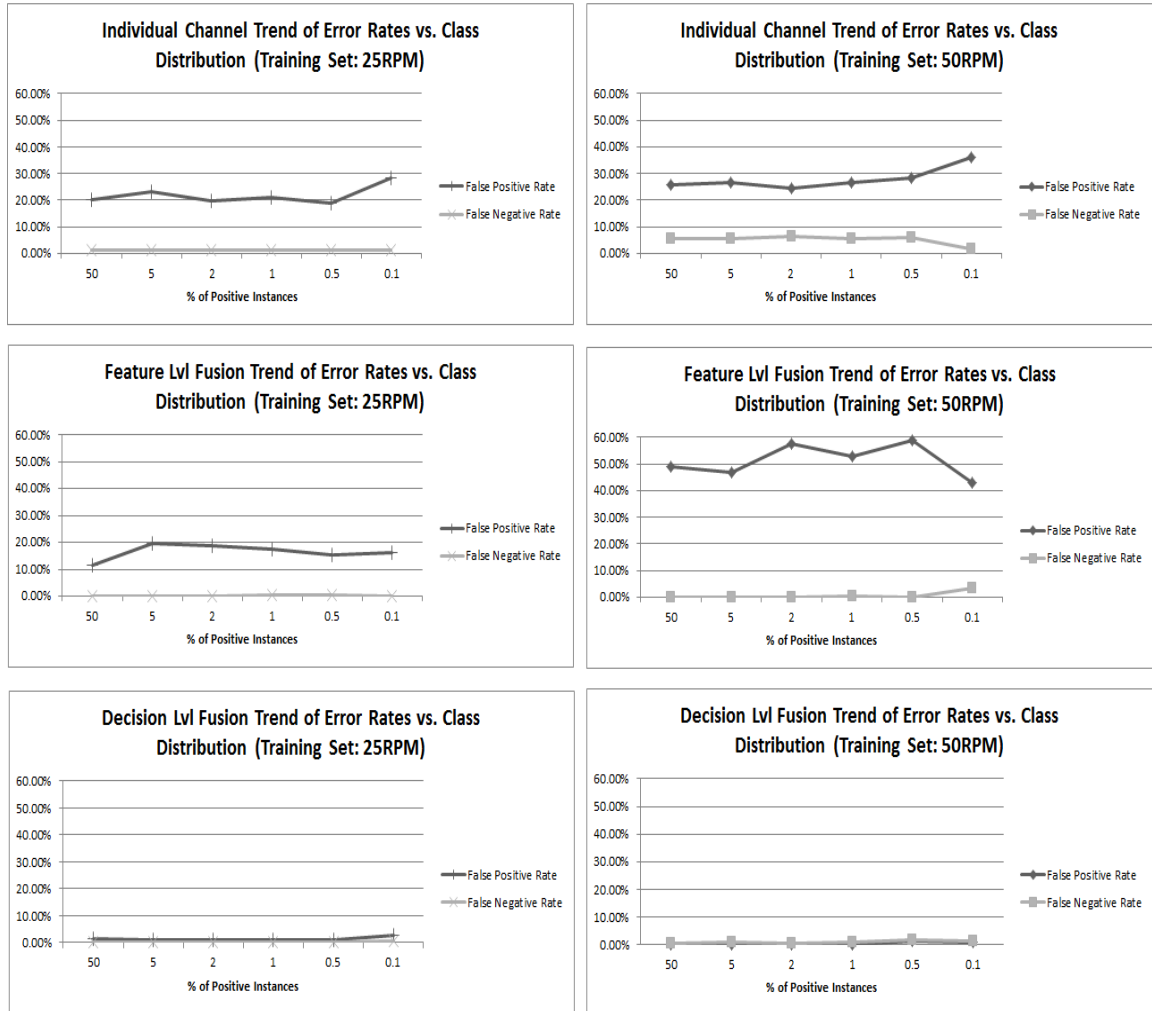


Figure 8.4: Error Rates

is consistent for both training sets and all class distributions. It is important to note, however, that the average FPR of models built from feature-level fused data was better than the FPR of those models built on the unfused data for the first case study (where the 25RPM data was used for training) but was significantly higher than the FPR of the models built on the unfused data for the second case study (where the training set was the 50RPM data). Based on the overall error rates and increased classification performances that are observed when decision-level fusion is applied, it can be concluded that this fusion technique is superior to feature-level fusion on this data.

8.7.4 Which learners perform best regardless of the speed at which the training data was collected?

The SVM and 5-NN perform consistently across both training sets. This is observed in Figures 8.1 and 8.2 where the representative clusters of bars for these two learners in the leftmost graphs are most similar to the corresponding clusters in the rightmost graphs.

8.7.5 What is the optimal combination of data fusion type (feature level or decision level) and learner?

Overall, we find the 5-NN and MLP models are the most robust to class distribution when trained on fused data (using either of the two fusion techniques and any RPM). Based on the observed error rates (Figure 8.4), the decision-level fusion approach is the preferred technique for combining the sensor data.

8.7.6 Analysis of Results

A three-way ANalysis of VAriance (ANOVA) F Test [61] was conducted on the AUC to statistically examine the effects of the various data fusion techniques and class distributions on the performances of the classification models. Each RPM (25RPM and 50RPM) was analyzed separately. An n -way ANOVA can be used to determine if the means in a set of data differ when grouped by multiple factors. If they do differ, additional tests (such as the Tukey’s Honestly Significant Difference criterion) can be used to determine which factors or combinations of factors are associated with the difference. The three factors in this study are:

- Factor A. Results per sensor fusion technique, *i.e.*, feature-level versus decision-level.
- Factor B. The seven learners.
- Factor C. The results for each of 6 class distributions as listed in Table 8.1.

Tables 8.2 and 8.3 show the ANOVA test results for the 25RPM and 50RPM datasets, respectively. A significance level of 5% ($\alpha = 0.05$) was utilized for all tests. This means that if the value in the rightmost column of each of the two tables equals or exceeds 0.05 for any individual factor or group of factors, then that factor or that group of factors produce similar classification performances and thus do not significantly impact the mean performances. From the data presented in these tables, we can reasonably conclude therefore that for both RPMs, the choice of learner, the dataset distribution and the fusion technique employed all significantly impact the results (both individually and jointly).

Tukey’s Honestly Significant Difference criterion was used to perform a multiple comparison test for each of the three factors to further analyze how the various

Source	Sum sq.	d.f.	Mean sq.	F	Prob < F
A	0.82665	1	0.82665	253.92	0
B	1.24802	6	0.208	63.89	0
C	0.04586	5	0.00917	2.82	0.0157
A \times B	1.06486	6	0.17748	54.52	0
A \times C	0.04383	5	0.00877	2.69	0.0201
B \times C	0.23463	30	0.00782	2.4	0
A \times B \times C	0.23162	30	0.00772	2.37	0.0001
Error	2.46117	756	0.00326		
Total	6.15664	839			

Table 8.2: ANOVA – 25RPM

Source	Sum sq.	d.f.	Mean sq.	F	Prob < F
A	11.2056	1	11.2056	1824.47	0
B	9.4951	6	1.5825	257.66	0
C	0.8367	5	0.1673	27.25	0
A \times B	8.7654	6	1.4609	237.86	0
A \times C	0.7824	5	0.1565	25.48	0
B \times C	4.5105	30	0.1503	24.48	0
A \times B \times C	4.3131	30	0.1438	23.41	0
Error	4.6432	756	0.0061		
Total	44.5521	839			

Table 8.3: ANOVA Results – 50RPM

values for each factor affect the overall results. A significance level of 5% ($\alpha = 0.05$) was again utilized for these tests. The results are presented as six graphs, shown in Figures 8.5(a) to 8.5(f), where the symbol \circ represents the group mean and the interval around that symbol is the 95% confidence interval. Intervals which are disjoint indicate a significant difference (given $\alpha = 0.05$); those which overlap imply no significant differences.

It is clear from Figures 8.5(a) and 8.5(b) that, as previously observed, decision-level fusion is drastically better than feature-level fusion in both case studies. There is a clear distinction between the mean values for both fusion techniques and no overlap of their confidence intervals.

From the comparison test conducted on Factor B (the machine learning algorithms), we find again that the k -Nearest Neighbor (5-NN) and Multi-Layer Perceptron (MLP) are consistently among the top three learners regardless of RPM. The decision tree (C4.5) was the worst performer. Although the Random Forest (RF) performed similar to the MLP and 5-NN when trained on the 25RPM data, there is a steep degradation of its performance if the 50RPM data is used for training. On the 50RPM data, the RF performs slightly better than (but statistically similar to) the Naive Bayes (NB). The Support Vector Machines (SVM) performs similarly to the 5-NN and MLP on the 50RPM training set and slightly worse on the 25RPM training set.

Finally, analysis of the effects of Factor C confirms that the class distribution does impact classification performances. The mean performances on the balanced dataset (50% positive instances) for both RPMs are better than those on the datasets with the four highest degrees of skew. At 25RPM, datasets containing 5% positive instances result in significantly worse performances than the balanced datasets but at 50RPM, performances on the 5% imbalanced dataset are similar to (if not slightly better) than

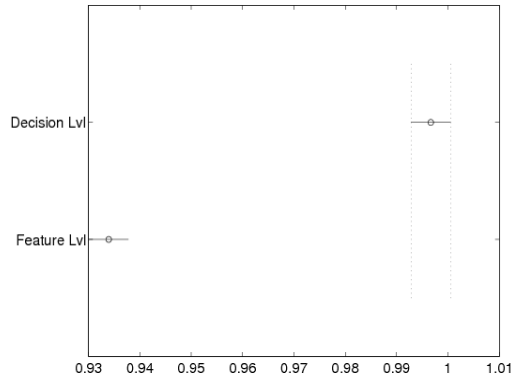
the results on the balanced dataset. With or without fusion, however, the datasets with 1%, 0.1%, 2% and 0.5% positive instances yield worst performances than the remaining two distributions (which have lesser degrees of imbalance) on the 50RPM data and all but the 1% and 5% distributions were statistically significantly worst than the balanced dataset.

8.8 CHAPTER SUMMARY

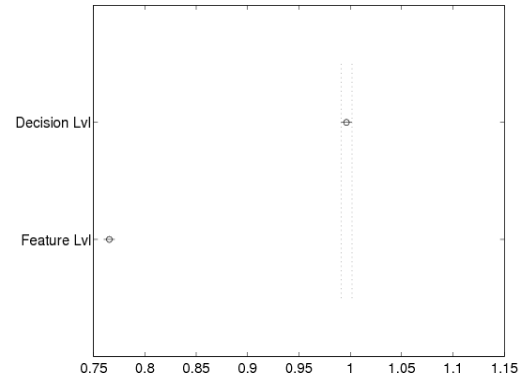
Imbalanced datasets pose a problem for traditional machine learners. Much research effort has been invested into understanding and alleviating class imbalance in many domains, but not much work has yet been done studying this problem in the context of condition monitoring and little or none of these are related to ocean turbine reliability. In experiments presented in this chapter, we gauged the effect of a skewed dataset for detecting ocean turbine state by training seven popular machine learning algorithms on datasets with varying class distributions. We also analyzed how different techniques for fusing the data from multiple sensor sources fared against class imbalance.

Findings from two case studies conducted on experimental data collected from an ocean turbine dynamometer indicated that the k -Nearest Neighbor machine learning algorithm performs extremely well on the given datasets regardless of the class distribution as well as the speed at which the training data was collected. This tolerance to the speed and other environmental condition of the dynamometer is especially important because ocean turbines are subject to an ever-changing operating environment. The ocean turbine's speed will also continuously vary during operations, and an automated state detection module must consistently perform well to avoid damage to the machine and unnecessary downtime. Classification models may need to be rebuilt

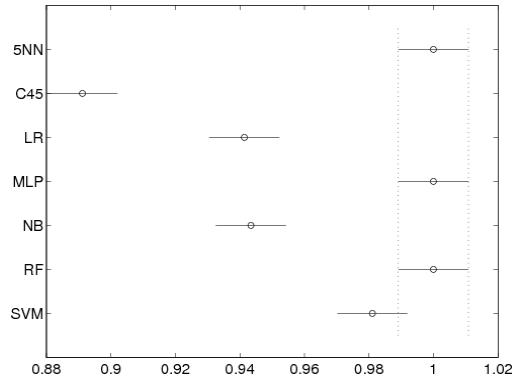
online because of a change in system state and thus the speed at which the training data was gathered should not be a factor which could negatively affect the models' ability to distinguish among the system states. Although the k -Nearest Neighbor models built on individual sensor data are satisfactory, perfect models can be generated from that learner by applying the decision-level fusion approach described in this paper. Thus, the combination of vibration analysis (the Short Time Wavelet Transform with Baselining technique), decision-level fusion approach and either the k -Nearest Neighbor or Multi-Layer Perceptron produce an excellent candidate model for an ocean turbine state detection module.



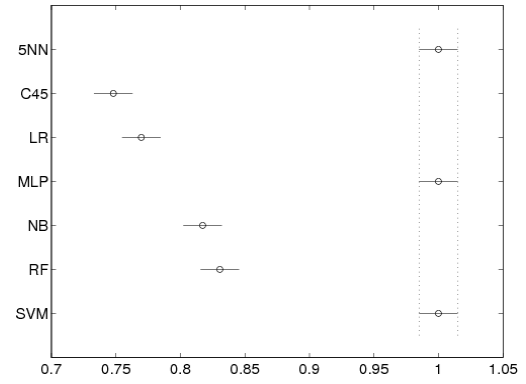
(a) Factor A (Fusion Technique) – 25RPM



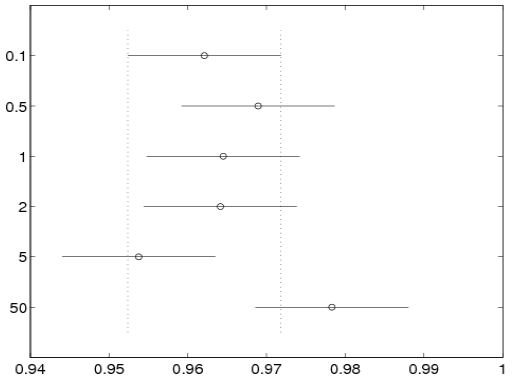
(b) Factor A (Fusion Technique) – 50RPM



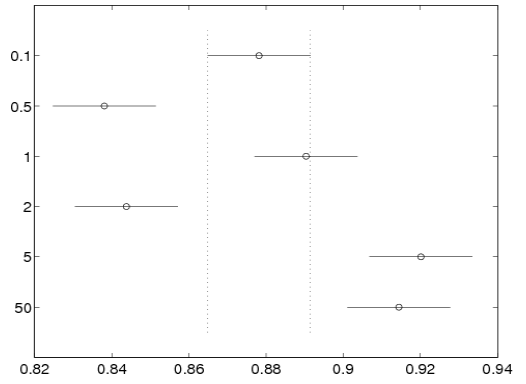
(c) Factor B (Learner) – 25RPM



(d) Factor B (Learner) – 50RPM



(e) Factor C (Class Distribution) – 25RPM



(f) Factor C (Class Distribution) – 50RPM

Figure 8.5: Multiple comparisons on the 3 three main factors

Chapter 9

Condition Monitoring Software System (CMSS)

9.1 INTRODUCTION

Numerous reliability concerns mostly attributed to the unpredictable nature of an ocean turbine's operating environment motivate the need for an automated monitoring system to ensure these turbines satisfy uptime and productivity requirements and to reduce operating costs. Machine condition monitoring (MCM) systems provide such a means for continuous and intelligent problem detection while Prognostics and Health Monitoring (PHM) systems work to predict the useful life and life expectancy of these machines. MCM/PHM systems employ a network of sensors to record data about the behaviour and operational environment of the systems they monitor and utilize a wide variety of data mining or machine learning techniques, statistical and inferencing tools, waveform analysis methods and data fusion algorithms to derive useful information from the sensor data. By performing automated monitoring of the ocean turbine, potential faults can be detected and identified at an early stage allowing for quick remediation, such as self adjustment or shutdown, to minimize damage to the turbine, thus reducing downtime due to failure.

The primary contribution of this chapter is the Condition Monitoring Software

System (CMSS) tool, a software tool for monitoring ocean turbines and other submerged vessels, which is presented in Section 9.2. In its current state, the CMSS tool performs data manipulation and state detection – two basic requirements of a condition based monitoring system per the International Organization for Standardization (ISO) standard ISO-13374 [74]. This data mining based tool is being developed to aid engineers in fault identification (determining what the problem is), fault localization (finding the location or source of the problem), fault prognosis (predicting the likely outcome from a problem) and future life assessment for an ocean turbine while it operates autonomously offshore. The completed product will be a data-driven MCM/PHM system in which sensor data is processed, fused and mined to understand the current behavior of the turbine and predict its future health. In the present version of the tool, a Short Time Wavelet Transform based technique is applied to raw vibration signals to transform the data into a format that classifiers can learn from. The outputs from n classification models built separately on data from each of the n sensor channels are combined using a decision-level data fusion approach to yield a single prediction of the current state of the machine based on all of the available sensor data. Classification models constructed using either the Multi-Layer Perceptron (MLP) or k -Nearest Neighbor algorithm were found to be optimal for this application. The planned data flow through this MCM/PHM system, whose architecture was so designed to conform with industry standards, is mapped out in this work. We also show how different operational states (such as changes in the counter-torque or load applied on the generator) could be reliably identified using this software tool.

9.2 CMSS

9.2.1 Requirements

The requirements of the tool are as follows:

1. The MCM/PHM architecture should use open standards. The PHM architecture should follow the OSA-CBM specification for defining data and interface specifications for communication between subsystems.
2. The MCM/PHM architecture must support remote configuration of its runtime parameters.
3. The MCM/PHM software will be deployed on a computer running the LINUX operating system.
4. The MCM/PHM architecture must be scalable and easily expandable to monitor multiple turbines
5. The MCM/PHM architecture must be easy to maintain and upgrade
6. The MCM/PHM architecture should permit remote monitoring of its operational status and performance.
7. The MCM/PHM architecture should allow deployment of new/updated subsystems remotely (from the shore system) and without taking the topside subsystems offline.
8. Although there are no real-time requirements (since critical system events are typically captured by the Safety Controller or LabVIEW Vibrations Monitoring subsystems shown in Figure 2.5), software must be as computationally efficient as possible so as to avoid a performance bottleneck. With thousands of vibration

measurements being produced per sensor every second, it is necessary to process each burst of data (or data stream as the case may be) quickly to prepare for the next batch of data.

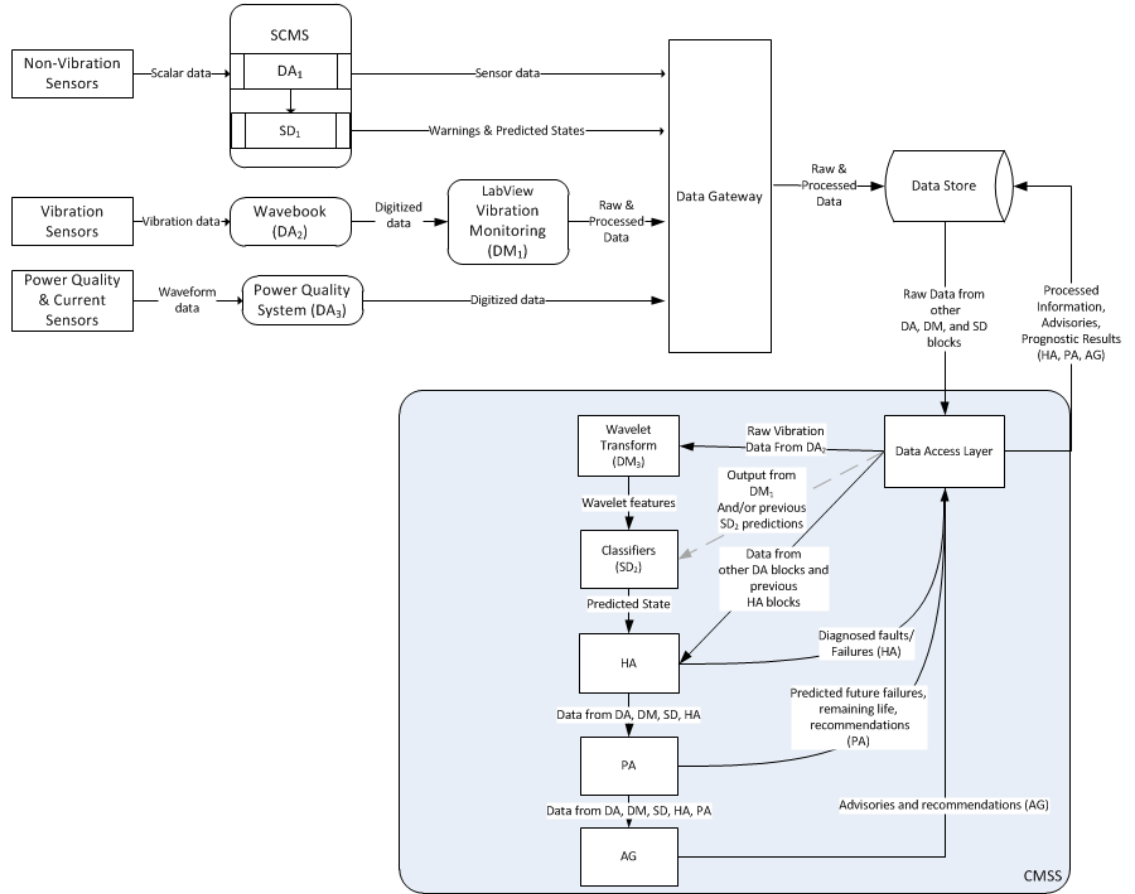
9.2.2 Architecture

Figure 9.1 shows the data flow through the software components of the MCM/PHM system. We present this view of the system to show the roles different sub-systems play in terms of the six ISO-13374 functions. These functions are:

1. Data Acquisition (DA) – Data is collected and digitized.
2. Data Manipulation (DM) – Typically, in the DM block, signal processing, time synchronous averaging (TSA), algorithmic computations and feature extraction are performed on the digitized output from the DA block.
3. State Detection (SD) – The output from DM and DA are compared against the anticipated baseline profile values to evaluate the system’s state in terms of a predefined enumeration, e.g., system normal, level high, alarm, alert, etc.
4. Health Assessment (HA) – Diagnosing system faults and determining the health of the system occurs in this block. The output from the DA, DM and SD blocks are fused with the output from other HA blocks in order to make this assessment.
5. Prognostics Assessment (PA) – The life expectancy and future health of the system are projected, and the possibility of future faults and failures is predicted.
6. Advisory Generation (AG) – Reports on existing or predicted conditions along with advice on how to optimize the life of the machine are generated.

Data acquisition is performed by three controllers. Vibration signals are recorded and pre-processed by a Wavebook Data Acquisition Unit (DA_2), power quality data via the Power Quality System (DA_3) and all remaining signals are acquired and handled by the Safety Control and Monitoring System, or SCMS (DA_1). The DA block has been split into these sub-blocks since they work independently of each other. By labeling the sub-systems according to role, it is now clear that there are multiple DA, DM and SD blocks (and not just one of each as was depicted in Figure 2.5) which all feed the MCM/PHM system. This highlights the need for data fusion.

Figure 9.1: Data Flow through MCM/PHM system



Two DM processes will operate on the vibration signals. The first is the LabVIEW Vibration Monitoring (VM) application [105] designed to conduct cepstrum and spectral analysis of the data. The VM application will be situated between the Wavebook and the Data Gateway to allow it real time (or as close to that as possible) access to the vibration signals as they are made available. Any system issues detected by this application will automatically trigger an alarm, alert or shutdown request depending on the severity of the problem. The VM application will then package the raw data, the results of the analyses, a history of any warnings/alerts/alarms and the parameters pertaining to the data acquisition (including the sampling rate, burst size, a unique identifier representing the sensor channel which generated the data and a 14 character MIMOSA identifier indicating which turbine the data was recorded from) as an XML file which is then passed to the Data Gateway to be timestamped and forwarded to the Data Store. Once this XML file arrives at the Data Store, the information contained within it is parsed and stored.

The second process is another vibration analysis technique applied to the raw vibration data to extract useful features (or descriptive characteristics) that can be used for data mining, data fusion and machine learning within the MCM/PHM system. Based on the outcome of previous experiments conducted by our team on data from the ocean turbine’s dynamometer, we recommend the use of a modified Short Time Wavelet Transform algorithm with Baseline-Differencing (STWTB) proposed by Wald *et al.* [153]. This is a two step process where the first step is the application of a Short Time Wavelet Transformation (STWT) algorithm. The second step is the use of “baseline-differencing” to normalize the data relative to a given operating condition. This baselining step was deemed necessary to remove those portions of the vibration signals that are characteristic of the machine’s environment and/or operational conditions (in this case, its rotational velocity), so that the remaining signal

only depicts the vibrations caused by actual abnormalities in the machine (and not its operating conditions). Prior applications of this algorithm produced good results as shown in our related work [47, 33] and in Chapters 7 and 8.

The output of the STWTB is a set of wavelet features, where the number of features is determined by the depth of the transform and the amount of data being examined at a given instance. Each feature represents the amount of oscillations in the vibration signal at a given wavelength. This STWTB output is supplied to state detection sub-block SD_2 where the state of the machine is again assessed. Within SD_2 , data fusion is required to combine the views of the n vibration sensors that are mounted on the turbine. For this, we recommend the decision level fusion approach employed in [33]. SD_1 refers to the state detection performed by the SCMS in which sensor values are compared against operational limits for that sensor type. If the sensor value is outside of the normal operating range, the SCMS will declare an abnormal state and may trigger an alarm or alert depending on the severity of the problem.

The output from the DA, DM and SD blocks as well as earlier HA results are then fed into the HA block so that faults or failures can be diagnosed and the likely source of any problem(s) can be determined. These advisories and data are then forwarded to the PA and AG blocks for prognostics and advisory generation. Future studies will investigate the use of Expert Systems [95], Fuzzy learning [96] and/or other machine learning techniques for combining the available information and for performing the required tasks of the HA block. We will also experiment with different approaches for prognostics and advisory generation.

9.2.3 Implementation

The CMSS application was designed according to the Model-View-Controller (MVC) pattern, a popular and widely accepted software design pattern used to separate the way the data is processed (Model) from visual presentation (View). The Controller serves as an intermediary between the Model and View, passing commands back and forth. The application was coded in the JAVA programming language to maximize portability and maintain extensibility. JAVA is an object oriented language, meaning that each entity within an application is coded as its own unit, known as a class. Each entity can therefore be reused and the application can be modularized so that it can be easily extended. Also, unlike many languages, JAVA is supported on most if not all operating systems (including LINUX), and programs written in JAVA can therefore be easily ported to a computer running a different operating system if the need arises.

Either of two popular machine learning algorithms, Multi-Layer Perceptron (MLP) [14] or k -Nearest Neighbors (k -NN) [55] was determined via experimentation [33] to be ideal for the SD₂ block. These algorithms were found to be robust to class imbalance and both performed extremely well regardless of the current environmental condition when used in conjunction with STWTB and a decision-level data fusion approach. The k -NN algorithm classifies a new instance by taking a majority vote of the classes of the k instances in the training dataset that are closest to the new instance within the feature space. The MLP is a form of feed-forward neural network which maps input values to an output. It uses a learning technique known as back-propagation, a generalization of the least mean squares algorithm, which involves continually updating the weights it assigns to individual connections within the neural network based on the amount of error in the output compared to the expected outcome.

The decision level data fusion [36] method involves first generating a classification model from the data from each sensor channel separately. For each new incoming instance, the probability that this new instance belongs to each of the possible system states is then computed individually based on observations made by each source. These probabilities are averaged and the system state with the highest probability is selected as the fused output.

CMSS references an open source data mining library WEKA [160] – the engine behind the popular WEKA data mining tool¹ – in which the k -NN and MLP along with many other algorithms are implemented. Default values were selected for all parameters of the IBk algorithm (the WEKA implementation of the k -NN algorithm) with the exception of the value of k which was set to 5 and the *distanceWeighting* (which tells the learner to use an inverse distance weighting to determine how to classify an instance) set to ‘Weight by 1/distance’. For the MLP, two parameters were changed: the *hiddenLayers* parameter (the number of nodes in the intermediate, or hidden, layers in the network) was set to ‘3’ and the *validationSetSize* (percentage of the training dataset reserved for validating the MLP model during back-propagation) was set to ‘10’.

The output from the LabView Vibration Monitoring (VM) system is currently the only input to the CMSS. The LabView VM can save each batch (or burst) of data to either a standardized format that the Data Gateway will be able to interpret or to a LabView Measurement File (LVM) with each column of data being readings from a different sensor channel and each row being the raw vibration amplitudes read at a particular time interval. Both input formats are considered acceptable to the CMSS Data Access Layer. Regardless of the original input format of the vibration signals, the Data Access Layer will re-format this data and save this to the file format

¹Available for download from <http://www.cs.waikato.ac.nz/ml/weka>

expected by the module that performs the STWTB transformation. Note that the STWTB Data Manipulation function is also implemented separately.

The typical use case based on the functionality that is currently implemented is as follows:

1. Training data from n sensor channels are used to construct n MLP or 5-NN models for distinguishing between the m different states. This is done only once since these models are reapplied to the new data as they arrive. In the live system, it is possible that models may need to be rebuilt midstream.
2. Vibration data from the dynamometer testbed is collected and time synchronous averaged by the Wavebook Unit. The Wavebook Unit is currently tethered directly to the LabView VM.
3. The LabView VM performs cepstrum analysis of the data and aside from the analysis results, it also produces an LVM file with n column representing the time-amplitude readings from the n vibration sensors. The filename contains a date and time stamp of the data collection time.
4. The CMSS reads this n column file and invokes the STWTB function with current parameters, including the depth of the transform. The depth of the transform dictates the number of wavelet features that will be output by the algorithm and is constrained by the burst size. CMSS waits for completion of the STWTB algorithm, which produces n comma separated files per burst.
5. Once the STWTB process has ended, CMSS will read all n files and deserialize the data into n *Instances* objects – each represents a different dataset in the WEKA framework.

6. For the i^{th} Instances object, the i^{th} MLP or 5-NN is applied. The model is applied to each instance j and produces a degree of certainty that instance j belongs to each possible state. For each instance j , we have $m \times n$ probabilities: the probability given by each of the n models for each of the m classes/states. The class with the highest average probability across all the models is selected as the final decision for that instance.
7. CMSS results are written to a text file and presented on screen to the user. In the live system, these results will be transmitted either to the Data Store if the CMSS and Data Store reside shore-side or to the Data Gateway if the CMSS resides topside.

9.3 CONCLUSION

This chapter presented the architecture for a Condition Monitoring Software System (CMSS) tool – a data driven MCM/PHM system for monitoring ocean turbines and other submerged machines. The CMSS tool is designed to conform with ISO standards for condition monitoring systems to ensure interoperability with software written by other vendors. In its present state, CMSS performs state detection from vibration signals acquired by multiple sensors. The process it uses to transform, manipulate and interpret these signals is outlined in this chapter.

Conclusion and Future Work

Machine Condition Monitoring / Prognostics Health Monitoring (MCM/PHM) systems help to ensure the reliability of the machines they monitor by automating detection, identification and localization of problems soon after or even before they develop. Such systems employ numerous types of sensors, including vibration sensors, to record different physical phenomena of the machine being monitored. This research focuses on the design and development of an MCM/PHM system for ocean turbines. Many factors including remoteness, a harsh and unpredictable environment and high retrieval costs pose reliability concerns for these turbines.

10.1 CONCLUSIONS

Per ISO-13374, an international standard for designing a condition based monitoring system, there are six primary functions of an MCM/PHM system – data acquisition, data manipulation, state detection, health assessment, prognostics and advisory generation. Data fusion, which is the aggregation of data from multiple sources to gain a complete, more accurate view, is known to be needed to combine data in different ways to realize these functions and has been identified in our previous publications [39, 38] as a cross cutting concern. Chapter 5 addresses this by offering a data fusion framework for MCM/PHM systems which satisfies those ISO standards. Also pro-

posed in that chapter is a barrier synchronization approach for coordinating sensor streams within this framework.

Preliminary tests of data mining and data fusion techniques for accomplishing state detection from vibration data are presented in Chapter 6. These experiments, conducted on vibration data acquired from two sensors mounted on a household box fan, revealed that a feature-level fusion approach (where data fusion is applied after descriptive features are extracted from the raw data) produced the most consistent performances. Of the twelve classifiers in that study, 5-Nearest Neighbor seemed to give the best results, even more so when either feature-level or decision-level fusion is applied.

The streaming wavelet transform (SWT) used to extract features from each vibration signal would sometimes output vectors where some of the values had not yet been computed depending on the amount of data available at the time the transform is applied. We evaluate how this missingness affects the performances of each learner in one of three case studies presented in Chapter 7. Data for all three studies were acquired from six sensors mounted on a dynamometer test platform for the ocean turbine. It was observed that the 5-Nearest Neighbor, which performed extremely well on vibration data from the fan, is the worst performer of the seven classifiers used in this study, although the data were pre-processed using the same SWT algorithm. Upon application of an Expectation-Maximization Imputation (EMI) or Mean Imputation (MI) technique to fill in the missing values, the 5-Nearest Neighbor now produces excellent models. The Decision Tree seemed to perform best on the SWT processed vibration data from the dynamometer, most likely due to repeated values produced by that algorithm, with and without data imputation. Also of note is the effectiveness of the quick and easy MI imputation technique which replaces the missing values with the mean of the available values as compared to the more sophisticated

EMI counterpart.

The last two studies in that chapter revolved around feature selection (FS) – a popular pre-processing methodology for reducing the dimensionality of a dataset often resulting in more robust learning models. The first FS case study involved seven feature selection techniques and seven machine learners. Features were extracted using the SWT algorithm and fused using the feature-level approach that was previously identified as yielding satisfactory results. The performances of models built using the top 3, 4, 5, 7 and 10 features identified by each FS technique were analyzed and compared against those of models built using all features. The second FS study compared the performances of four machine learners when trained on all features against their performances when trained on a subset of 2, 4, 6, 8, 10 and 15 features selected by five different feature selection techniques.

This second study differs from the first in the way models are evaluated. The observed performances of the learners in the first study assumed that new, incoming data would share similar characteristics as those data used to build the models. This is highly unlikely in a live ocean turbine system as its environmental conditions and operating parameters, like its rotational velocity, continuously fluctuate during operation. In this second FS study, we test models on data collected while the dynamometer operated at a different speed than the one at which the data used to build the models was collected. We selected STWTB as the vibration analysis (and feature extraction) approach of choice for the third study in that chapter based on findings from a previous study [151]. Further justification for this choice is presented in the published paper [46]. In both studies, the models built from 5-Nearest Neighbors (5-NN) and Logistic Regression (LR) yielded the best results. These results show how using just a subset of wavelet transform features extracted from the vibration signal, we can achieve similar or better results than when all features are used. This

is particularly true for the 5-NN, Naive Bayes (NB) and LR algorithms with the 5-NN and LR learners generating perfect models from just 8 and 2 features respectively. The NB model was also near perfect when trained on just the top 8 features selected using the PRC feature ranking technique.

A scenario that must also be accounted for is that, in the dataset used to build our models, there will be significantly fewer examples of problem states than there will be of normal operation. Experiments in Chapter 8 investigate how varying degree of class imbalance will affect various learners. These experiments extend one of our published studies [37] which performed a similar analysis but at a much smaller scale. Findings confirm that the combination of the STWTB vibration analysis method, decision-level fusion approach and either the 5-Nearest Neighbor or Multi-Layer Perceptron produce an excellent candidate model for an ocean turbine state detection module.

The Condition Monitoring Software System (CMSS) tool proposed in Chapter 9 and presented in our related work [34] brings the entire process together. Presented in that chapter are specifics of how the proposed data-driven MCM/PHM approach for this ocean turbine prototype could be realized and implemented.

10.2 FUTURE WORK

Opportunities for future work include:

- In all of these experiments, only binary classification (distinguishing between two states) was considered. Multi-class classification remains for future work.
- Missing from the current implementation of the CMSS are the health assessment, prognostics and advisory generation processes which remain. Next steps would involve experimentation with various data mining, machine learning and data fusion strategies for conducting health assessment and prognostics and

the integration of these remaining modules (health assessment, prognostics and advisory generation) into the CSS.

- Data fusion approaches employed in these studies assume that all sources are equally reliable and that the data or information being combined emanated from the same type of format (and hence exists in the same format). As there are many different types of sensors and data sources that would comprise a complete MCM/PHM system, appropriate fusion techniques must be employed to combine the data at different points within the system. Evaluation of other fusion algorithms such as those listed in a survey paper compiled by our team [41] is therefore also necessary future work.
- It will also be necessary to evaluate said vibration analysis, data fusion and data mining techniques on vibration data gathered from the live ocean turbine. At the time our research was conducted, the ocean turbine was still under development.

BIBLIOGRAPHY

- [1] Agena Ltd. Agenarisk software package, 2011.
- [2] D. Aleksendric and D. C. Barton. Neural network prediction of disc brake performance. *Tribology International*, 42(7):1074–1080, 2009.
- [3] W. Altidor, T. Khoshgoftaar, and J. V. Hulse. Robustness of filter-based feature ranking: A case study. In *Proceedings of the Florida Artificial Intelligence Research Society (FLAIRS’11) Conference*, 2011.
- [4] P.-P. Beaujean, T. M. Khoshgoftaar, J. C. Sloan, N. Xiros, and D. Vendittis. Monitoring Ocean Turbines: A Reliability Assessment. In *Proceedings of the 15th ISSAT International Reliability and Quality in Design Conference*, pages 367–371, 2009.
- [5] R. Bhowmik. Data mining techniques in fraud detection. *Journal of Digital Forensics, Security and Law*, 3(2), 2008.
- [6] H. Boudali and J. B. Dugan. A continuous-time bayesian network reliability modeling, and analysis framework. *IEEE Transactions on Reliability*, 55(1):86–97, 2006.
- [7] L. Breiman. Random Forests. In *Machine Learning*, pages 5–32, 2001.
- [8] S. Budhaditya, D.-S. Pham, M. Lazarescu, and S. Venkatesh. Effective anomaly detection in sensor networks data streams. In *Ninth IEEE International Conference on Data Mining, 2009. ICDM ’09*, pages 722–727, Dec, 2009.
- [9] M. D. Buhmann and M. D. Buhmann. *Radial Basis Functions*. Cambridge University Press, New York, NY, USA, 2003.
- [10] C. S. Byington and A. K. Garga. Data Fusion for Developing Predictive Diagnostics for Electromechanical Systems. In M. E. Liggins, D. L. Hall, and J. Llinas, editors, *Handbook of Multisensor Data Fusion – Theory and Practice, Second Edition*, chapter 28, pages 701–737. CRC Press, City, State or Country, 2008.
- [11] Y. D. Cai, D. Clutter, G. Pape, J. Jan, M. Welge, and L. Auvil. Maids: Mining alarming incidents from data streams. *Proceedings of the 23rd ACM SIGMOD*

- International Conference on Management of Data*, pages 13–18, Jun. 2004. Paris, France.
- [12] I. Cardei, A. Agarwal, B. Alhalabi, T. Tavtilov, T. Khoshgoftaar, and P.-P. Beaujean. Software and communications architecture for Prognosis and Health Monitoring of ocean-based power generator. In *IEEE International Systems Conference (SysCon), 2011*, pages 353–360, Apr. 2011.
 - [13] S. Ceccherini, B. Carli, U. Cortesi, S. D. Bianco, and P. Raspollini. Retrieval of the vertical column of an atmospheric constituent from data fusion of remote sensing measurements. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 111(3):507 – 514, 2010.
 - [14] D. Charalampidis and B. Muldrey. Clustering using multilayer perceptrons. *Nonlinear Analysis: Theory, Methods & Applications*, 71(12):e2807 – e2813, 2009.
 - [15] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.
 - [16] G. Chen, J. Xu, and X. Xiang. Neighborhood Preprocessing SVM for Large-Scale Data Sets Classification. In *Proceedings of the 2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery - Volume 02*, FSKD ’08, pages 245–249, Washington, DC, USA, 2008. IEEE Computer Society.
 - [17] K. Chen, L. Wang, and H. Chi. Methods of combining multiple classifiers with different features and their applications to text-independent speaker identification. *International Journal of Pattern Recognition and Artificial Intelligence*, 11:417–445, 1997.
 - [18] Q. Cheng, P. Varshney, J. Michels, and C. Belcastro. Distributed Fault Detection with Correlated Decision Fusion. *IEEE Transactions on Aerospace and Electronic Systems*, 45(4):1448–1465, Oct. 2009.
 - [19] S. Cho, S. Binsaeid, and S. Asfour. Design of multisensor fusion-based tool condition monitoring system in end milling. *The International Journal of Advanced Manufacturing Technology*, 46:681–694, 2010. 10.1007/s00170-009-2110-z.
 - [20] K. Choi, S. Singh, A. Kodali, and K. R. Pattipati. Novel Classifier Fusion Approaches for Fault Diagnosis in Automotive Systems. *IEEE Transactions on Instrumentation and Measurement*, pages 602–611, 2009.
 - [21] W. W. Cohen. Fast Effective Rule Induction. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123, 1995.

- [22] D. C. Conner, P. R. Kedrowski, and C. F. Reinholtz. Multiple camera, laser rangefinder, and encoder data fusion for navigation of a differentially steered 3-wheeled autonomous vehicle. In *Proceedings of SPIE, the International Society for Optical Engineering*, pages 76–83, 2001.
- [23] G. Cormode and S. Muthukrishnan. Summarizing and mining skewed data streams. In *Scientific Data Mining (SDM)*, 2005.
- [24] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos. Compressing historical information in sensor networks. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, pages 527–538, New York, NY, USA, 2004. ACM.
- [25] Y. Diao, D. Ganesan, G. Mathur, and P. J. Shenoy. Rethinking data management for storage-centric sensor networks. In *CIDR 2007, Third Biennial Conference on Innovative Data Systems Research*, pages 22–31, Asilomar, CA, USA, Jan. 2007.
- [26] P. Domingos and M. J. Pazzani. On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. *Machine Learning*, 29:103–130, 1997.
- [27] J. Don and Z. Wang. Effects of anti-oxidant migration on friction and wear of c/c aircraft brakes. *Applied Composite Materials*, 16:73–81, 2009. 10.1007/s10443-008-9075-1.
- [28] J. Dong, D. Zhuang, Y. Huang, and J. Fu. Advances in Multi-Sensor Data Fusion: Algorithms and Applications. *Sensors*, 9(10):7771 – 7784, 2009.
- [29] D. L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, pages 1289–1306, 2004.
- [30] F. R. Driscoll, G. M. Alsenas, P. P. Beaujean, S. Ravenna, J. Raveling, E. Bussold, and C. Slezycki. A 20 kW open ocean current test turbine. In *Proceedings of the MTS/IEEE Oceans '08*, pages 1–6, Quebec City, Quebec, Canada, Sept. 2008.
- [31] D. Dubois and H. Prade. Possibility Theory and Data Fusion in Poorly Informed Environments. *Control Engineering Practice*, 2(5):811–823, 1994.
- [32] J. Duhaney and T. M. Khoshgoftaar. Comparing Sensor Fusion Approaches for Ocean Turbine Monitoring and Reliability. Technical report, Department of Computer and Electrical Engineering and Computer Science at Florida Atlantic University, Boca Raton, FL, USA, 2010.

- [33] J. Duhaney and T. M. Khoshgoftaar. A Study on Class Imbalance in Ocean Turbine Fault Data. Technical report, Department of Computer and Electrical Engineering and Computer Science at Florida Atlantic University, Boca Raton, FL, USA, 2012.
- [34] J. Duhaney and T. M. Khoshgoftaar. CMSS: A Software Tool for Monitoring Ocean Turbines. Technical report, Department of Computer and Electrical Engineering and Computer Science at Florida Atlantic University, Boca Raton, FL, USA, 2012.
- [35] J. Duhaney and T. M. Khoshgoftaar. Fusing Wavelet Features for Ocean Turbine Fault Detection. Technical report, Department of Computer and Electrical Engineering and Computer Science at Florida Atlantic University, Boca Raton, FL, USA, 2012.
- [36] J. Duhaney and T. M. Khoshgoftaar. Decision level fusion of wavelet features for ocean turbine state detection. In *Eleventh International Conference on Machine Learning and Applications (ICMLA 2012)*, forthcoming 2012.
- [37] J. Duhaney and T. M. Khoshgoftaar. Studying the Effect of Class Imbalance in Ocean Turbine Fault Data on Reliable State Detection. In *Eleventh International Conference on Machine Learning and Applications (ICMLA 2012)*, forthcoming 2012.
- [38] J. Duhaney, T. M. Khoshgoftaar, A. Agarwal, and J. C. Sloan. Mining and Storing Data Streams for Reliability Analysis. In *Proceedings of the 16th ISSAT International Reliability and Quality in Design Conference*, pages 314 – 318, Washington D.C., USA, August 2010.
- [39] J. Duhaney, T. M. Khoshgoftaar, I. Cardei, B. Alhalabi, and J. C. Sloan. Applications of Data Fusion in Monitoring Inaccessible Ocean Machinery. In *Proceedings of the 16th ISSAT International Reliability and Quality in Design Conference*, pages 308 – 313, Washington D.C., USA, August 2010.
- [40] J. Duhaney, T. M. Khoshgoftaar, I. Cardei, B. Alhalabi, and J. C. Sloan. Data and Knowledge Fusion for MCM/PHM in Inaccessible Ocean Systems. Technical report, Department of Computer and Electrical Engineering and Computer Science at Florida Atlantic University, Boca Raton, FL, USA, 2010.
- [41] J. Duhaney, T. M. Khoshgoftaar, and J. C. Sloan. Data Fusion for Reliability Analysis: A Survey. Technical report, Department of Computer and Electrical Engineering and Computer Science at Florida Atlantic University, Boca Raton, FL, USA, 2011.

- [42] J. Duhaney, T. M. Khoshgoftaar, and J. C. Sloan. Feature level sensor fusion for improved fault detection in MCM systems for ocean turbines. In *Proceedings of the 24th Florida Artificial Intelligence Research Society Conference (FLAIRS'24)*, pages 15–20, 2011.
- [43] J. Duhaney, T. M. Khoshgoftaar, and J. C. Sloan. Feature Selection on Dynamometer Data for Reliability Analysis. In *Proceedings of the 23rd IEEE International Conference on Tools with Artificial Intelligence (ICTAI'11)*, pages 1012–1019, 2011.
- [44] J. Duhaney, T. M. Khoshgoftaar, and J. C. Sloan. A survey of data fusion algorithms for reliability analysis. In *Proceedings of the 17th ISSAT International Reliability and Quality in Design Conference*, pages 344–348, 2011.
- [45] J. Duhaney, T. M. Khoshgoftaar, J. C. Sloan, B. Alhalabi, and P. P. Beaujean. A Dynamometer for an Ocean Turbine Prototype – Reliability Through Automated Monitoring. In *Proceedings of the 13th IEEE International High Assurance Systems Engineering Symposium*, pages 244–251, Boca Raton, FL, USA, 2011.
- [46] J. Duhaney, T. M. Khoshgoftaar, and R. Wald. Applying feature selection to short time wavelet transformed vibration data for reliability analysis of an ocean turbine. In *Eleventh International Conference on Machine Learning and Applications (ICMLA 2012)*, forthcoming 2012.
- [47] J. Duhaney, T. M. Khoshgoftaar, R. Wald, and P. P. Beaujean. Fusion of Wavelet Transform Features for Reliable Fault Detection within an Ocean Turbine MCM System. In *Proceedings of the 18th ISSAT International Reliability and Quality in Design Conference*, pages 23 – 27, 2012.
- [48] J. Duhaney, T. M. Khoshgoftaar, R. Wald, and J. C. Sloan. Classifier Robustness to Missing Values – a Case Study in Ocean Turbine State Detection. Technical report, Department of Computer and Electrical Engineering and Computer Science at Florida Atlantic University, Boca Raton, FL, USA, 2011.
- [49] Z. Elouedi, K. Mellouli, and P. Smets. Assessing Sensor Reliability for Multi-sensor Data Fusion Within the Transferable Belief Model. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 34(1):782–787, Feb. 2004.
- [50] J. Esteban, A. Starr, R. Willetts, P. Hannah, and P. Bryanston-Cross. A review of data fusion models and architectures: towards engineering guidelines. *Neural Computing and Applications*, 14(4):273–281, 2005.

- [51] S. Fabre, A. Appriou, and X. Briottet. Sensor Fusion Integrating Contextual Information. *International Journal of Uncertainty, Fuzziness & Knowledge-Based Systems*, 9(3):369, 2001.
- [52] C. Fan, Z. Jin, J. Zhang, and W. Tian. Application of multisensor data fusion based on RBF neural networks for fault diagnosis of SAMS. In *Control, Automation, Robotics and Vision, 2002. ICARCV 2002. 7th International Conference on*, volume 3, pages 1557 – 1562, Dec. 2002.
- [53] W. Fan. Systematic data selection to mine concept-drifting data streams. *Proceedings of the SIGKDD04*, pages 128–137, 2004.
- [54] A. Folleco, T. M. Khoshgoftaar, J. V. Hulse, and L. A. Bullard. Software quality modeling: The impact of class noise on the random forest classifier. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2008*, pages 3853–3859, 2008.
- [55] R. Fraiman, A. Justel, and M. Svarc. Pattern recognition via projection-based kNN rules. *Computational Statistics & Data Analysis*, 54(5):1390–1403, 2010.
- [56] H. Freeman and O. Lowenschuss. Bibliography of sampled-data control systems and z-transform applications. *IRE Transactions on Automatic Control*, 4(1):28 – 30, Mar. 1958.
- [57] M. M. Gaber, S. Krishnaswamy, and A. Zaslavsky. Ubiquitous data stream mining. In *The Eighth Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Sydney, Australia, May 2004.
- [58] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy. Mining data streams: a review. *SIGMOD Record*, 34(2):18–26, 2005.
- [59] D. Ganesan, B. Greenstein, D. Perelyubskiy, D. Estrin, and J. Heidemann. An evaluation of multi-resolution storage for sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 89–102, New York, NY, USA, 2003. ACM.
- [60] K. Gao, T. M. Khoshgoftaar, and A. Napolitano. Stability of Filter-Based Feature Selection Methods for Imbalanced Software Measurement Data. In *Proceedings of the 24th International Conference on Software Engineering & Knowledge Engineering (SEKE'2012)*, pages 74–79, Jul. 2012.
- [61] K. Gao, T. M. Khoshgoftaar, H. Wang, and N. Seliya. Choosing software metrics for defect prediction: an investigation on feature selection techniques. *Software: Practice and Experience*, 41(5):579–606, 2011.

- [62] V. García, J. S. Sánchez, R. A. Mollineda, R. Alejo, and J. M. Sotoca. The class imbalance problem in pattern classification and learning. In F. J. F.-T. et al, editor, *II Congreso Español de Informática*, pages 283–291, Zaragoza, 2007. Thomson.
- [63] N. A. Giacobe. Application of the JDL data fusion process model for cyber security. In J. J. Braun, editor, *Multisensor, Multisource Information Fusion: Architectures, Algorithms, and Applications 2010*, volume 7710, page 77100R. SPIE, 2010.
- [64] A. H. Gunatilaka and B. A. Baertlein. Feature-level and decision-level fusion of noncoincidentally sampled sensors for land mine detection. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 23:577–589, 2001.
- [65] X. Guo, Y. Yin, C. Dong, G. Yang, and G. Zhou. On the class imbalance problem. In *Proceedings of the 2008 Fourth International Conference on Natural Computation - Volume 04*, ICNC '08, pages 192–201, Washington, DC, USA, 2008. IEEE Computer Society.
- [66] D. L. Hall and S. A. H. McMullen. *Mathematical Techniques in Multisensor Data Fusion (Artech House Information Warfare Library)*. Artech House Publishers, March 2004.
- [67] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: an update. *SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.
- [68] S. Hashemi, Y. Yang, Z. Mirzamomen, and M. Kangavari. Adapted one-versus-all decision trees for data stream classification. *IEEE Transactions on Knowledge and Data Engineering*, 21(5):624–637, 2009.
- [69] R. Herrtwich. Timed data streams in continuous-media systems. Technical Report TR-90-017, International Computer Science Institute, Berkley CA, May 1990.
- [70] C. Hiransoog and C. Malcolm. Multi-sensor/knowledge fusion. In *Proceedings of the 1999 IEEE/SICE/RSJ International Conference on Multisensor Fusion and Integration for Intelligent Systems, 1999. MFI '99.*, pages 117–122, Malcolm, CA, USA, Aug. 1999.
- [71] J. Huang and C. Ling. Using AUC and accuracy in evaluating learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 17(3):299 – 310, Mar. 2005.
- [72] J. V. Hulse and T. Khoshgoftaar. Knowledge discovery from imbalanced and noisy data. *Data and Knowledge Engineering*, 68(12):1513–1542, Dec 2009.

- [73] *IEEE Standard Glossary of Software Engineering Terminology*. USA, December 1990. IEEE Std 610.12-1990.
- [74] International Organization for Standardization. ISO 13374. Condition Monitoring and Diagnostics of Machines – Data Processing, Communication and Presentation.
- [75] N. Japkowicz. The Class Imbalance Problem: Significance and Strategies. In *In Proceedings of the 2000 International Conference on Artificial Intelligence (ICAI)*, pages 111–117, 2000.
- [76] M. A. K. Jaradat and R. Langari. A hybrid intelligent system for fault detection and sensor fusion. *Applied Soft Computing*, 9:415–422, Jan. 2009.
- [77] P. Jayaswal, S. Verma, and A. Wadhwani. Application of ann, fuzzy logic and wavelet transform in machine fault diagnosis using vibration signal analysis. *Journal of Quality in Maintenance Engineering*, 16(2):190–213, 2010.
- [78] A. D. Jurik and A. C. Weaver. Control, analysis and visualization of body sensor streams. In *International Symposium on Medical Information and Communication Technology (ISMICT)*, Montreal, QC, Canada, Feb. 2009.
- [79] A. Karlsson. Dependable and generic high-level information fusion – methods and algorithms for uncertainty management. Technical Report HS-IKI-TR-07-003, Institutionen för kommunikation och information, School of Humanities and Informatics, University of Skövde, Sweden, 2007.
- [80] T. Khoshgoftaar, L. Bullard, and G. K. Attribute Selection Using Rough Sets in Software Quality Classification. *International Journal of Reliability, Quality, and Safety Engineering*, 16(1):73 – 89, 2009.
- [81] T. Khoshgoftaar, S. Zhong, and V. Joshi. Enhancing Software Quality Estimation Using Ensemble-Classifer Based Noise Filtering. *Intelligent Data Analysis: An International Journal*, 6(1):3 – 27, 2005.
- [82] T. M. Khoshgoftaar, M. Golawala, and J. V. Hulse. An Empirical Study of Learning from Imbalanced Data Using Random Forest. In *ICTAI '07: Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence*, pages 310–317, Washington, DC, USA, 2007. IEEE Computer Society.
- [83] T. M. Khoshgoftaar and J. Hulse. Imputation techniques for multivariate missingness in software measurement data. *Software Quality Control*, 16:563–600, Dec. 2008.

- [84] T. M. Khoshgoftaar, L. A. Nguyen, K. Gao, and J. Rajeevalochanam. Application of an Attribute Selection Method to CBR-Based Software Quality Classification. In *Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'03)*, pages 47–52, Sacramento, CA, USA, Nov. 2003.
- [85] T. M. Khoshgoftaar and J. van Hulse. Imputation techniques for multivariate missingness in software measurement data. *Software Quality Control*, 16(4):563–600, 2008.
- [86] T. Kobayashi and D. L. Simon. Application of a Bank of Kalman Filters for Aircraft Engine Fault Diagnostics. Technical report, National Aeronautics and Space Administration (NASA), Cleveland OH, Glenn Research Center, 2003.
- [87] E. Kuljanic and M. Sortino. Twem, a method based on cutting forces–monitoring tool wear in face milling. *International Journal of Machine Tools and Manufacture*, 45(1):29–34, 2005.
- [88] M. Last. Online classification of nonstationary data streams. *Intelligent Data Analysis*, 6(2):129–147, 2002.
- [89] S. Le Cessie and J. C. Van Houwelingen. Ridge Estimators in Logistic Regression. *Applied Statistics*, 41(1):191–201, 1992.
- [90] M. Lebold, K. McClintic, R. Campbell, C. Byington, and K. Maynard. Review of vibration analysis methods for gearbox diagnostics and prognostics. *Proceedings of the 54th Meeting of the Society for Machinery Failure Prevention Technology*, pages 623–634, May 2000.
- [91] C.-H. Lee, C.-R. Lin, and M.-S. Chen. Sliding-window filtering: an efficient algorithm for incremental mining. In *Proceedings of the Tenth International Conference on Information and Knowledge Management, CIKM '01*, pages 263–270, New York, NY, USA, 2001. ACM.
- [92] X. Li, P. S. Yu, B. Liu, and S.-K. Ng. Positive unlabeled learning for data stream classification. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2009*, pages 257–268, Sparks, Nevada, USA, 2009.
- [93] C. X. Ling and V. S. Sheng. Cost-sensitive Learning and the Class Imbalanced Problem. In C. Sammut, editor, *Encyclopedia of Machine Learning*. Springer, 2011.
- [94] L. Liu, A. Wang, M. Sha, X. Sun, and Y. Li. Optional SVM for Fault Diagnosis of Blast Furnace with Imbalanced Data. *ISIJ International*, 51(9):1474–1479, 2011.

- [95] S. Liu and S. Liu. An efficient expert system for machine fault diagnosis. *International Journal of Advanced Manufacturing Technology*, 21(9):691–698, 2003.
- [96] X. Liu, L. Ma, and J. Mathew. Rotating machinery fault diagnosis based on fuzzy data fusion techniques. In *2nd World Congress on Engineering Asset Management and the 4th International Conference on Condition Monitoring*, pages 1309–1318, Harrogate, England, Jun. 2007.
- [97] Y. Liu. Imbalanced text classification: A term weighting approach. *Expert Systems with Applications*, 36(1):690–701, 2009. doi:10.1016/j.eswa.2007.10.042; pmid:.
- [98] Z. Liu, D. Forsyth, J. Komorowski, K. Hanasaki, and T. Kirubarajan. Survey: State of the Art in NDE Data Fusion Techniques. *IEEE Transactions on Instrumentation and Measurement*, 56(6):2435 – 2451, Dec. 2007.
- [99] J. Llinas, C. Bowman, G. Rogova, A. Steinberg, E. Waltz, and F. White. Revisiting the JDL Data Fusion Model II. In *Proceedings of the Seventh International Conference on Information Fusion (FUSION 2004)*, pages 1218–1230, 2004.
- [100] Z. Ma and A. Survival. Survival analysis approach to reliability, survivability and prognostics and health management (phm). In *Aerospace Conference, 2008 IEEE*, pages 1 –20, Mar. 2008.
- [101] T. Marwala. Introduction to missing data. *Computational Intelligence for Missing Data Imputation, Estimation, and Management: Knowledge Optimization Techniques*, pages 1 – 18, 2009.
- [102] M. Mazzucco, A. Ananthanarayan, R. L. Grossman, J. Levera, and G. B. Rao. Merging multiple data streams on common keys over high performance networks. In *Supercomputing '02: Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, pages 1–12, Los Alamitos, CA, USA, 2002. IEEE Computer Society Press.
- [103] MIMOSA. OSA-CBM V3.3.0, 2010.
- [104] Minerals Management Service. Technology White Paper on Ocean Current Energy Potential on the U.S. Outer Continental Shelf, May 2006.
- [105] M. Mjit. Methodology for fault detection and diagnostics in an ocean turbine using vibration analysis and modeling. Masters, Florida Atlantic University, 777 Glades Road, Boca Raton FL 33431, 2009.
- [106] Z. P. Mourelatos and J. Zhou. Reliability Estimation and Design with Insufficient Data Based on Possibility Theory. In *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, volume 43, pages 1696 – 1705, 2004.

- [107] M. Neil, N. Fenton, S. Forey, and R. Harris. Using Bayesian Belief Networks to Predict the Reliability of Military Vehicles. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems (IJUFKS)*, Special issue on management of uncertainty and imprecision in multimedia information systems, 12(1):11–20, 2001.
- [108] M. Neil, N. Fenton, and D. Marquez. Using Bayesian Networks and Simulation for Data Fusion and Risk Analysis. *NATO Science for Peace and Security Series: Information and Communication Security*, 13, 2007.
- [109] M. Neil, P. Krause, and N. E. Fenton. Software quality prediction using bayesian networks. In T. M. Khoshgoftaar, editor, *Software Engineering with Computational Intelligence*, chapter Chapter 6. Kluwer, 2003.
- [110] T. M. Nguyen, J. Schiefer, and A. M. Tjoa. Sense & response service architecture (SARESA): an approach towards a real-time business intelligence solution and its use for a fraud detection application. In *DOLAP '05: Proceedings of the 8th ACM international workshop on Data warehousing and OLAP*, pages 77–86, New York, NY, USA, 2005. ACM.
- [111] G. Niu, T. Han, B.-S. Yang, and A. C. C. Tan. Multi-agent decision fusion for motor fault diagnosis. *Mechanical Systems and Signal Processing*, 21(3):1285 – 1299, 2007.
- [112] C. Olston, J. Jiang, and J. Widom. Adaptive filters for continuous queries over distributed data streams. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 563–574, New York, NY, USA, 2003. ACM.
- [113] M. E. Orlowska, X. Sun, and X. Li. Can exclusive clustering on streaming data be achieved? *SIGKDD Explorations Newsletter*, 8(2):102–108, 2006.
- [114] R. Orsagh, J. Sheldon, and C. Klenke. Prognostics/diagnostics for gas turbine engine bearings. In *Proceedings of the 2003 IEEE Aerospace Conference*, volume 7, pages 3095–3103, Mar. 2003.
- [115] M. Oussalah, H. Maaref, and C. Barret. Application of a possibilistic-based approach to mobile robotics. *Journal of Intelligent and Robotic Systems*, 38:175–195, Oct. 2003.
- [116] R. Pan. A Unifying Approach to Data Fusion for Reliability Prediction. In *Proceedings of Mathematical Methods in Reliability (MMR 2009)*, pages 173 – 177, June 2009.

- [117] C. Parikh, M. Pont, and N. Jones. Application of Dempster-Shafer Theory in Condition Monitoring Applications: A Case Study. *Pattern Recognition Letters*, 22(6-7):777–785, May 2001.
- [118] Z. K. Peng and F. L. Chu. Application of the wavelet transform in machine condition monitoring and fault diagnostics: a review with bibliography. *Mechanical Systems and Signal Processing*, 18(2):199–221, 2004.
- [119] C. S. Penrod and T. J. Wagner. Another Look at the Edited Nearest Neighbor Rule. *IEEE Transactions on Systems, Man and Cybernetics*, 7(2):92–94, Feb. 1977.
- [120] P. Pirjanian, J. A. Fayman, and H. I. Christensen. Improving Task Reliability by Fusion of Redundant Homogeneous Modules Using Voting Schemes. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 425–430, 1997.
- [121] J. C. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. *Advances in Kernel Methods: Support Vector Learning*, 208(MSR-TR-98-14):1–21, 1998.
- [122] J. R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.
- [123] D. Raheja, J. Llinas, R. Nagi, and C. Romanowski. Data fusion/data mining-based architecture for condition-based maintenance. *International Journal of Production Research*, 44(14):2869–2887, Jul. 2006.
- [124] A. Rao and D. Jones. A denoising approach to multisensor signal estimation. *IEEE Transactions on Signal Processing*, 48(5):1225–1234, May 2000.
- [125] A. Rehman, D. R. Phalke, and R. Pandey. Alternative fuel for gas turbine: Esterified jatropha oil-diesel blend. *Renewable Energy*, 36(10):2635 – 2640, 2011. Renewable Energy: Generation & Application.
- [126] D. Ruta and B. Gabrys. An Overview of Classifier Fusion Methods. *Computing and Information Systems*, 7:1 – 10, 2000.
- [127] J. Rutten. A tutorial on coinductive stream calculus and signal flow graphs. *Theoretical Computer Science*, 343(3):443 – 481, 2005. Formal Methods for Components and Objects.
- [128] F. Samadzadegan. Data Integration Related to Sensors, Data and Models. In *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences (ISPRS)*, number 4 in 35, pages 569–574. Natural Resources Canada, 2004.

- [129] J. Sargolzaei and A. Kianifar. Modeling and simulation of wind turbine savonius rotors using artificial neural networks for estimation of the power ratio and torque. *Simulation Modelling Practice and Theory*, 17(7):1290 – 1298, 2009.
- [130] N. Seliya, T. M. Khoshgoftaar, and J. V. Hulse. A study on the relationships of classifier performance metrics. In *Proceedings of the 2009 21st IEEE International Conference on Tools with Artificial Intelligence*, ICTAI '09, pages 59–66, Washington, DC, USA, 2009. IEEE Computer Society.
- [131] G. Shafer. Belief functions. In *Readings in Uncertain Reasoning*. Morgan Kaufmann Publishers, 1990.
- [132] J. Sherman, R. Davis, W. Owens, and J. Valdes. The autonomous underwater glider "Spray". *Oceanic Engineering, IEEE Journal of*, 26(4):437 – 446, Oct. 2001.
- [133] M. L. Sichitiu and C. Veerarittiphan. Simple, accurate time synchronization for wireless sensor networks. In *IEEE Wireless Communications & Networking Conference*, 2003.
- [134] P. Smets. The transferable belief model. *Artificial Intelligence*, 66:191–234, Apr. 1994.
- [135] P. Smets. Data fusion in the transferable belief model. In *Proceedings of the International Conference on Information Fusion*, pages 21–33, Paris, France, Jul. 2000.
- [136] D. Smith and S. Singh. Approaches to Multisensor Data Fusion in Target Tracking: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 18(12):1696–1710, 2006.
- [137] W. Staszewski. Wavelet Based Compression and Feature Selection for Vibration Analysis. *Journal of Sound and Vibration*, 211:735–760, 1998.
- [138] A. N. Steinberg, C. L. Bowman, and F. E. White. Revisions to the JDL Data Fusion Model. *Sensor Fusion: Architectures, Algorithms, and Applications, Proceedings of the SPIE*, 3719(1):430–441, 1999.
- [139] M. Stonebraker, U. Çetintemel, and S. Zdonik. The 8 requirements of real-time stream processing. *SIGMOD Record*, 34(4):42–47, 2005.
- [140] X. Su, T. M. Khoshgoftaar, and R. Greiner. Making an accurate classifier ensemble by voting on classifications from imputed learning sets. *International Journal of Information and Decision Sciences (IJIDS)*, pages 301–322, 2009.
- [141] Z. Sun and C. C. Chang. Structural damage assessment based on wavelet packet transform. *Journal of Structural Engineering*, 128(10):1354–1361, 2002.

- [142] Y. Tian, G. Weiss, D. Frank, and H. Q. Ma. A combinatorial fusion method for feature mining. In *Proceedings of KDD'07 Workshop on Mining Multiple Information Sources*, pages 6–13, 2007.
- [143] I. Tomek. Two Modifications of CNN. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-6(11):769–772, Nov. 1976.
- [144] J. Triesch and C. V. D. Malsburg. Democratic integration: Self-organized integration of adaptive cues. *Neural Computation*, 13(9):2049–2074, 2001.
- [145] L. G. Valiant. A bridging model for parallel computation. *Communications of the ACM*, 33(8):103–111, 1990.
- [146] J. Van Hulse, T. Khoshgoftaar, and N. A. Experimental perspectives on learning from imbalanced data. In *Proceedings of the 24th International Conference on Machine Learning - ICML 2007*, Corvallis, OR, USA, Jun. 2007.
- [147] J. Van Hulse and T. M. Khoshgoftaar. A comprehensive empirical evaluation of missing value imputation in noisy software measurement data. *Journal of Systems and Software*, 81(5):691–708, 2008.
- [148] J. H. VanZwieten, W. E. Laing, and C. R. Slezycki. Efficiency assessment of an experimental ocean turbine generator. In *Proceedings of the MTS/IEEE Oceans Conference*, Kona, Hawai'i, U.S.A., Sep. 2011. IEEE Ocean Engineering Society.
- [149] K. Veeramachaneni, L. Osadciw, A. Ross, and N. Srinivas. Decision-level fusion strategies for correlated biometric classifiers. In *Proceedings of IEEE Computer Society Workshop on Biometrics at the Computer Vision and Pattern Recognition (CVPR) conference*, pages 1–6, Anchorage, AK, USA, 2008.
- [150] A. Verma and A. Kusiak. Fault Monitoring of Wind Turbine Generator Brushes: A Data-Mining Approach. *Journal of Solar Energy Engineering*, 134(2):021001, 2012.
- [151] R. Wald, T. M. Khoshgoftaar, and B. Alhalabi. A novel baseline-differencing approach for creating generalizable reliability models of ocean turbine behavior. In *Proceedings of the 18th ISSAT International Reliability and Quality in Design Conference*, pages 1–5, Jul. 2012.
- [152] R. Wald, T. M. Khoshgoftaar, P.-P. Beaujean, and J. C. Sloan. Combining wavelet and fourier transforms in reliability analysis of ocean systems. In *Proceedings of the 16th ISSAT International Reliability and Quality in Design Conference*, pages 303 – 307, Washington D.C., USA, 2010.

- [153] R. Wald, T. M. Khoshgoftaar, P.-P. J. Beaujean, and J. C. Sloan. A review of prognostics and health monitoring techniques for autonomous ocean systems. In *Proceedings of the 16th ISSAT International Reliability and Quality in Design Conference*, pages 308–313, Aug. 2010.
- [154] R. Wald, T. M. Khoshgoftaar, and J. C. Sloan. Fourier transforms for vibration analysis: A review and case study. In *The 12th IEEE International Conference on Information Reuse and Integration (IRI'11)*, pages 366–371, 2011.
- [155] D. Wang, M. Altar, and R. Sampson. An experimental investigation on cavitation, noise, and slipstream characteristics of ocean stream turbines. *Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy*, 221:219 – 231, 2007.
- [156] H. Wang, W. Fan, P. S. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 226–235, New York, NY, USA, 2003. ACM.
- [157] H. Wang, T. M. Khoshgoftaar, and N. Seliya. How many software metrics should be selected for defect prediction? In R. C. Murray and P. M. McCarthy, editors, *FLAIRS Conference*. AAAI Press, 2011.
- [158] H. Wang, J. Yin, J. Pei, P. S. Yu, and J. X. Yu. Suppressing model overfitting in mining concept-drifting data streams. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 736–741, New York, NY, USA, 2006. ACM.
- [159] N. Wiener. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. The MIT Press, 1964.
- [160] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, 2 edition, 2005.
- [161] R. Wolff, K. Bhaduri, and H. Kargupta. A Generic Local Algorithm for Mining Data Streams in Large Distributed Systems. *IEEE Transactions on Knowledge and Data Engineering*, 21(4):465 –478, Apr. 2009.
- [162] J. Ye, K. Chen, T. Wu, J. Li, Z. Zhao, R. Patel, M. Bae, R. Janardan, H. Liu, G. Alexander, and E. Reiman. Heterogeneous data fusion for alzheimer’s disease study. In *KDD '08: Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1025–1033, New York, NY, USA, 2008. ACM.

- [163] C. Zao, E. hui Zheng, H. wei Xu, and L. Chen. Cost-sensitive multi-class svm with reject option: A method for steam turbine generator fault diagnosis. *International Journal of Computer Theory and Engineering (IJCTE)*, 3(1):77–83, 2011.
- [164] S. Zhang, Z. Jin, and X. Zhu. Missing data imputation by utilizing information within incomplete instances. *Journal of Systems and Software*, 84:452–459, Mar. 2011.
- [165] X.-M. Zhao, X. Li, L. Chen, and K. Aihara. Protein classification with imbalanced data. *Protein Science*, 70:1125–1132, 2008.