Master of Science Thesis in Electrical Engineering
Department of Electrical Engineering, Linköping University, 2021

Information Fusion of Data-Driven Engine Fault Classification from Multiple Algorithms

Ninos Baravdish

LINKÖPING UNIVERSITY

Master of Science Thesis in Electrical Engineering

Information Fusion of Data-Driven Engine Fault Classification from Multiple Algorithms

Ninos Baravdish

LiTH-ISY-EX--21/5402--SE

Supervisor: **Max Johansson**
ISY, Linköpings universitet

Examiner: **Daniel Jung**
ISY, Linköpings universitet

Division of Vehicle Systems
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping, Sweden

Sammanfattning

I takt med att bilindustrin ständigt gör tekniska framsteg ställs allt högre krav på säkerhet, miljövänlighet och hållbarhet. Moderna fordon är på väg mot alltmer komplexa system, både när det kommer till hårdvara och mjukvara, vilket gör det viktigt att detektera fel i någon av komponenterna. Övervakning av motorns tillstånd har traditionellt gjorts med hjälp av expertkunskaper och modellbaserade metoder, där härledda modeller av systemets nominella tillstånd används för att detektera eventuella avvikelser. På grund av systemets ökade komplexitet möter detta tillvägagångssätt dock begränsingar vad gäller tid och kunskap för att beskriva motorns tillstånd. Ett alternativ är därför datadrivna metoder som istället baseras på historiska data uppmätt från olika arbetspunkter som används för att dra slutsatser om motorns nuvarande tillstånd.

I den här studien presenteras ett föreslaget diagnosramverk som består av ett systematiskt tillvägagångssätt för felklassificering av kända och okända fel samt en felstorleksuppskattning. Grunden för detta ligger i att använda principalkomponentanalys för att hitta felvektorn för varje felklass och avkoppla ett fel i taget, vilket skapar olika underrum. En viktig del i det här arbetet har varit att undersöka effektiviteten i att ta hänsyn till flera klassificerare vid beslutsfattandet ur ett prestandaperspektiv. Aggregering av flera klassificerare görs genom att lösa ett kvadratiskt optimeringsproblem. För att utvärdera prestandan har en jämförelse gjorts med en random forest klassificerare.

Utvärdering med utmanande testdata visar lovande resultat där algorithmen förhåller sig väl prestandamässigt med random forest klassificerare.

Abstract

As the automotive industry constantly makes technological progress, higher demands are placed on safety, environmentally friendly and durability. Modern vehicles are headed towards increasingly complex system, in terms of both hardware and software making it important to detect faults in any of the components. Monitoring the engine's health has traditionally been done using expert knowledge and model-based techniques, where derived models of the system's nominal state are used to detect any deviations. However, due to increased complexity of the system this approach faces limitations regarding time and knowledge to describe the engine's states. An alternative approach is therefore data-driven methods which instead are based on historical data measured from different operating points that are used to draw conclusion about engine's present state.

In this thesis a proposed diagnostic framework is presented, consisting of a systematically approach for fault classification of known and unknown faults along with a fault size estimation. The basis for this lies in using principal component analysis to find the fault vector for each fault class and decouple one fault at the time, thus creating different subspaces. Importantly, this work investigates the efficiency of taking multiple classifiers into account in the decision making from a performance perspective. Aggregating multiple classifiers is done solving a quadratic optimization problem. To evaluate the performance, a comparison with a random forest classifier has been made.

Evaluation with challenging test data show promising results where the algorithm relates well to the performance of random forest classifier.

Acknowledgments

I would like to thank my supervisor Max Johansson for all support and guidance in this work. I would also like to thank my examiner Daniel Jung who with continuous feedback and support played a major roll in the completion of this project.

Last but not least I would like to thank my family for being by my side all the way with lots of support and encouragement.

Linköping, June 2021
Ninos Emanuel Baravdish

Contents

Notation **xi**

1 Introduction **1**
- 1.1 Motivation . . . 2
  - 1.1.1 Data analysis . . . 2
  - 1.1.2 Weigh Fault Hypotheses . . . 2
  - 1.1.3 Fault Diagnosis System . . . 3
- 1.2 Aim and Purpose . . . 3
- 1.3 Research questions . . . 3
- 1.4 Delimitations . . . 4

2 Theory **5**
- 2.1 Multiple Classifier System . . . 5
  - 2.1.1 Static Selection . . . 7
  - 2.1.2 Dynamic Selection . . . 7
- 2.2 Fault Diagnosis . . . 8
  - 2.2.1 Fault Diagnosis - Important Concepts . . . 9
- 2.3 Principal Component Analysis . . . 9
- 2.4 K-Nearest Neighbor . . . 11
- 2.5 Support Vector Machines . . . 13
- 2.6 Error Correcting Output Codes . . . 16

3 Method **19**
- 3.1 Decoupling of fault using PCA . . . 19
- 3.2 Data Processing by Reshaping . . . 24
- 3.3 Classifier Selection . . . 25
- 3.4 Information Fusion . . . 27
  - 3.4.1 Local Classifier Weigthing by Quadratic Programming - Multi-Class Classifier . . . 27
  - 3.4.2 Local Classifier Weigthing by Quadratic Programming - Binary Classifier . . . 29
- 3.5 Fault Size Estimation . . . 31
- 3.6 Complete Diagnostic Framework . . . 31

3.6.1 Preparation 32
3.6.2 Classification 33
3.6.3 Evaluation 33

4 Results **35**
4.1 Data Collection 35
4.2 Preprocessing 38
4.3 Training Classifiers 40
4.3.1 Binary Classifiers 40
4.3.2 Multi-Class Classifiers 42
4.4 Data Classification 43
4.4.1 Binary Classifiers 44
4.4.2 Multi-Class Classifiers 48
4.4.3 Comparing with Random Forest 51
4.4.4 Fault Size Estimation 53
4.5 Preprocessing Data by Reshape 54
4.5.1 Binary Classifiers - Prediction 54
4.5.2 Multi-Class Classifiers - Prediction 55

5 Discussion **57**
5.1 Results 57
5.2 Methodology 59

6 Conclusion **61**
6.1 Future Work 62

Bibliography **63**

Notation

Mathematical Notation

Notation	Meaning
Ω	A set $\{\omega_1, ..., \omega_L\}$ of L class labels
C	A set $\{c_1, ..., c_M\}$ of M base classifiers
$\mathbf{X}$	An $r \times c$ rectangular matrix, where it is assumed that $r \gg c$. $\mathbf{X}$ represents residual data, where each row represents an observation and each column represents a residual
x_q	A test sample with an unknown class label

Abbrevations

Abbrevation	Meaning
MCS	Multiple Classifier System
POC	Pool of Classifiers
EOC	Ensemble of Classifiers
OCC	One-Class Classifier
MCC	Multi-Class Classifier
SVD	Singular Value Decomposition
PCA	Principal Component Analysis
SVM	Support Vector Machine
KNN	K-Nearest Neighbor
ECOC	Error Correcting Output Codes
MSE	Mean Squared Error

1 Introduction

Autonomous vehicles have advanced significantly over the past years and along with it they have become more complex than ever before. However, they also require higher demand on safety and reliability while on the roads. It is therefore crucial for the driver to be alerted when there is a fault in one of the components, since it might lead to reduced functionality in the vehicle or in the worst case a non-functional component. A result of any of these cases might lead to expensive reparation costs or a danger to the driver and passengers or even the surroundings. This is where diagnostics systems in autonomous vehicles comes in to reduce and prevent such events.

In today's combustion engines it is possible to measure relevant quantities in different components with the help of sensors, in order to monitor the engine's health. Thanks to the access to large amount of data, machine learning and data-driven classification have become increasingly useful and important. With residuals that are based on measured data from different fault sensors, it is possible to determine when an alarm has been detected and then draw conclusions about the engine's health. However, the residuals could be strongly correlated since they are based on the same sensor signals. This means that the information from all residuals cannot simply be added up, since they could partly give the same information if they are correlated.

The purpose of this thesis is to present a diagnosis system for an internal combustion engine (ICE) that combines multiple fault classification algorithms and performs an information fusion. The procedure is intended to carefully classify residual data that is correlated between different class labels. In Chapter 1, the studied problem is described along with the aim and purpose. Chapter 2 presents the theoretical background to the methodology in Chapter 3 but also a brief introduction about the studied area. Proceeding with Chapter 3, the proposed method is described in detail and evaluation of it from experimental results are followed by Chapter 4. Lastly, a discussion regarding the experimental results and the methodology are found in Chapter 5 and a final conclusion about the work in Chapter 4.

1.1 Motivation

This section describes the studied problems in this thesis from a general point of view, but also some of the related research topics.

1.1.1 Data analysis

Working with data-driven methods requires a lot of data, and since the models often are general it is beneficial to have good training data that represents the classes over relevant scenarios. However, an important consideration is whether all data really is good and for that matter necessary.

When dealing with fault diagnostics for an ICE, the work of [15] explains that one complicated aspect is the coupling between intake and exhaust flow through the turbine and compressor. The implication of this is that fault in any component is not isolated, but rather has the risk of affecting the performance of other components and thus affecting the sensor outputs elsewhere in the engine. Residual data generated from the same sensor output in the engine may lead to correlated predictors, which are then used to train classifiers.

One question that arises from this is whether each new data set brings new relevant information to the table, but also how the much relevant information the predictors have whitin the data set. From an economic and time perspective, it would be desirable to reduce the dimension on the data to lower the complexity and to get rid of unnecessary sensors that does not contribute to new information.

1.1.2 Weigh Fault Hypotheses

There are numerous methods to take multiple machine learning models into consideration in the decision making when predicting new data [3], [21], [24]. However, not all application areas share the same effectiveness on all methods and therefore there is a need to choose an appropriate one that fits the data. Taking advantage of multiple classifiers in pattern recognition tasks have nowadays lead to constructing a complete framework which includes different stages, such as classifier generation, classifier selection and integration. Along with these comes even more possibilities to adapt a custom made multiple classifier system, nevertheless only the imagination and development of technology sets limits to what is possible.

In traditional model-based fault diagnostics, decoupling of a fault is achieved when the fault is not sensitive to a certain test quantity. Hence, a similar approach inspired by such idea is yet a mystery to investigate when working with data-driven approaches. Weighing fault hypotheses from different classifiers trained on separately cases where one fault is decoupled from the rest opens many interesting possibilities. One of which is determining how important each classifier is and how much contribution it brings in the decision making.

1.1.3 Fault Diagnosis System

Along with the automotive industry development comes increasingly complexity, especially in the software part in the vehicle. Looking at the fault diagnostics aspect, it is headed towards being fueled by measured data from various sensors. This opens up new possibilities which allows to make more accurate predictions on for instance where a fault originates from based on data.

Some of the key components when constructing a functional diagnostic framework that relies on generated data requires robustness and reliability. Especially in an environment where it is meant to be used by a mechanic at a workshop, since troubleshooting the engine is common and necessary. Identifying the cause of an error message would not only save time, but also potentially save money for the car owner.

1.2 Aim and Purpose

The purpose of this thesis is to develop a diagnostic system for an ICE, that combines information from multiple fault classifiers. Using residual data to train different fault patterns, the goal is to extract sufficient information from multiple fault classifiers in order to draw a final diagnosis statement about the engine's state. It is desirable to minimize the use of data while maintaining the same diagnostic performance. The basis for this, which is a prerequisite, is to analyse the residual data and see what diversity and similarity there is amongst the residuals, but also between data sets from different faults in the engine.

1.3 Research questions

To sort out the problems describes in Section 1.1, a division has been made, in a divide-and-conquer fashion, according to each question at issue; analyse the data, fault hypothesis weighing and information fusion. Thus, the following research questions are formulated:

1. Analysing the residual data with fault vectors, i.e. the line along which data from a fault with different fault magnitudes lies within. To interpret how properties such as the angles between fault vectors or residuals affect the

classification performance. Furthermore, to investigate the possibility and how easy it is to decouple and isolate a fault with fault vectors along with minimizing the redundant of information.

2. How to weigh all fault hypotheses from different fault classifiers, and investigate how much relevant and new information classifier i contributes given that classifier j has stated its fault hypothesis.

3. How to construct a multiple classifier fusion algorithm for fault diagnosis. To combine and weigh fault hypotheses from multiple fault classifiers and form a final diagnosis statement about the engine's health.

1.4 Delimitations

To focus the study on the research questions stated above, the following delimitations were set:

- The given residual data comes from a previous work [15], where the amount of residuals are limited.
- The measurements has been collected from predetermined components in the engine, where each injected fault has limited span of fault size.
- The given residual data is assumed to be labelled after each corresponding fault type and fault size.

2 Theory

In this chapter, the fundamental concepts of the thesis are presented. Firstly, terms regarding the area for understanding how multiple data-driven approaches can be combined are presented, followed by a brief overview of some relevant terms regarding fault diagnosis and lastly the underlying mathematical definitions for this thesis.

2.1 Multiple Classifier System

Multiple classifier system (MCS) has become increasingly useful and popular in machine learning and pattern recognition over the past decades [6], [20], [28]. The reason for this is because it has shown to outperform single classifiers over various applications. By combining multiple classifiers that are diverse [1], rather than having one strong single classifier, has its advantage in obtaining higher classification accuracies. An illustration of a MCS can be seen in Figure 2.1 below.

[1] Diversity (or complementary) between classifiers refers to classifiers recognize different patterns, thus making independent classification errors

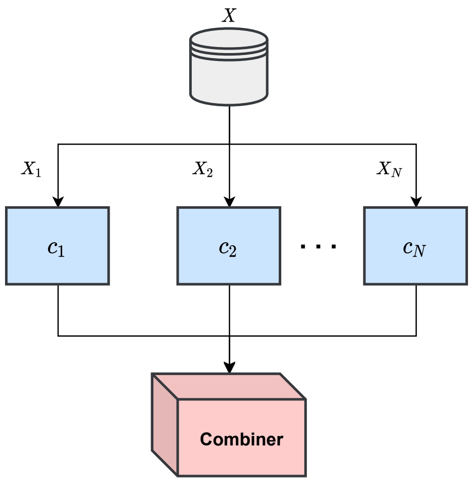


Figure 2.1: *An illustration of how a MCS is conducted.* **X** *represents the data that covers the entire feature space where* $\mathbf{X_1}, ..., \mathbf{X_N}$ *are subsets,* $C = \{c_1, ..., c_N\}$ *is the set of base learners and lastly a combiner that evaluates the outputs from the base learners and fuses the received information to form a final decision.*

In [6] it is explained that a MCS is composed by three stages: (1) *Generation*, (2) *Selection* and (3) *Fusion*. In the generation stage, a pool (or a set) of classifiers are trained such that they are diverse and have good accuracy[2]. It is also explained in the mentioned article that there are several different ways to generate trained classifiers. The more diversity and representation of the training data that these classifiers can capture, the better results in fused prediction will be achieved. The top three strategies mentioned are:

1. Different feature sets.
2. Different training sets.
3. Different classifier models.

Where it is stated that the first listed is more likely to generate a successful combination of classifiers. However, the article also refers to [26] which describes that a combination of strategies can be used together. For instance, to train classifiers with different feature sets and training data.

The selection stage attempts to choose a single classifier or an ensemble of classifiers (EOC) from the generated pool that is/are most competent. Lastly, the fusion stage is based on aggregating the decision outputs to give a final decision of the system.

[2]A rather vaguely measure, but since the accuracy differ for different applications it is up to the user to decide.

The articles [18] and [30] explains that there are generally two different approaches of combining multiple classifiers: *classifier fusion* and *classifier selection*. Classifier fusion is based on that all classifiers are trained over the entire feature space and their outputs are combined to achieve a form of group consensus. On the other hand, classifier selection assumes that each classifier is an expert in a subset of the feature space where it attempts to predict which of all single classifiers is most likely to conclude the correct diagnosis statement. In this project, the classifier selection is studied further.

The classifier selection stage can either be done in a *static* or *dynamic* way, which are explained below.

2.1.1 Static Selection

In this method, an ensemble of the most competent classifiers, C', is selected already during the training phase. The selection is based on certain criteria that is estimated in the validation data set. Essentially, the most competent classifiers are fixed after the selection. An illustration of the procedure can be seen in Figure 2.2 below.

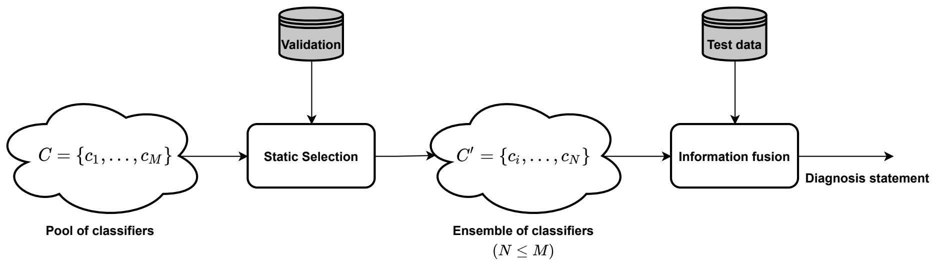


Figure 2.2: *Classifier selection in a static manner.*

2.1.2 Dynamic Selection

In this type of approach, it is assumed that the structure of the classifier ensemble, C', varies for each new incoming test sample. It is also assumed that each respective base classifier in C is locally competent in its own area. To define this region space, *k-nearest neighbour* is a common method [6]. Likewise in static selection, the most competent classifiers shall be determined, either a single classifier or an EOC.

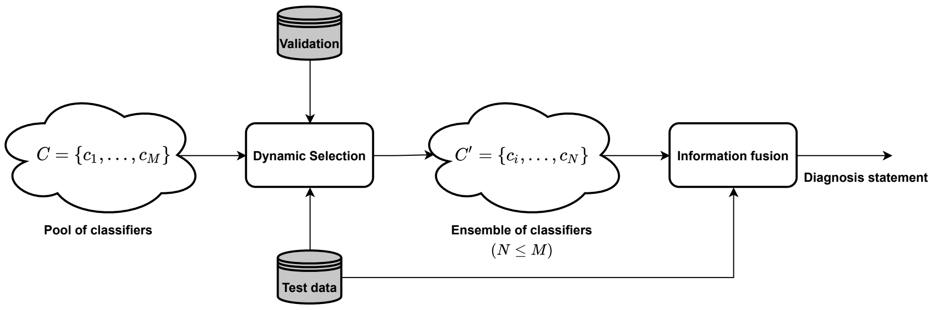


Figure 2.3: *Classifier selection in a dynamic manner.*

2.2 Fault Diagnosis

Considering a process, the general concept of a diagnosis system is to generate a diagnosis based on the knowledge of observed variables, in order to decide whether there is fault or not and to explicitly identify where it originates from. In the case where the diagnosis is based on models that tries to describe a technical system, e.g. an ICE, it is refered as *model-based diagnosis*.

Fault diagnosis has nowadays becoming more commonly used in industrial systems. It is about monitoring the system's state and detecting occurring faults by comparing model predictions of the systems nominal behaviour with measured sensor data on the monitored system. There are generally two different approaches for fault diagnosis: model-based and data-driven fault diagnosis. The basic principle in model-based fault diagnosis is to describe the system's nominal behaviour with a mathematical model. Any inconsistencies with the model predictions and sensor data is captured by residuals (the error between them) and detect a fault in the system [15], [19]. Figure 2.4 shows a generally view of how it works.

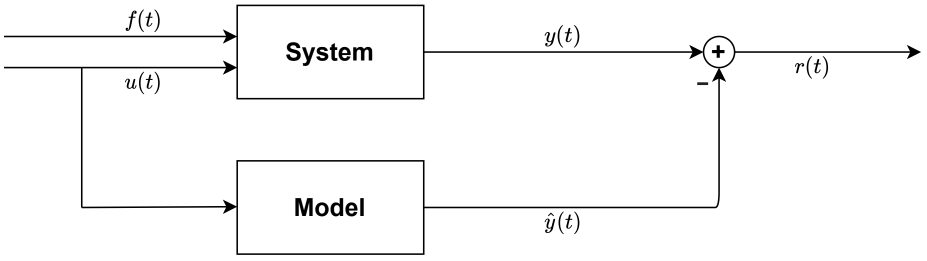


Figure 2.4: *Illustration of model-based fault diagnosis. Here, $f(t)$, $u(t)$, $y(t)$, $\hat{y}(t)$, $r(t)$ denotes the fault signal, actuator signal, output from the system, the modeled prediction and residual respectively at given time t during the measurement.*

Data-driven fault diagnosis on the other hand constructs models that relies on training data from different operating points and fault scenarios from the system. This, in order to capture how a set of input generates an output, where the output could for instance be the corresponding class label for the input data.

2.2.1 Fault Diagnosis - Important Concepts

Working with fault diagnosis there are some key concepts that are useful when investigating how different faults interact. The following definitions comes from [15].

Definition 2.1 (Fault sensitivity). A residual r_k is said to be sensitive to a fault f_i if $f_i \neq 0$ implies that $r_k \neq 0$. On the contrary, if there is a residual r_k that is not sensitive to fault f_i, then it is said to be *decoupled* in that specific residual.

Definition 2.2 (Fault isolation). A fault f_i is isolable from another fault f_j ($f_j \neq f_i$) if there is a residual r_k that is sensitive to f_i but not f_j.

Example 2.3: Illustrating basic terms in fault diagnosis

Assume an arbitrary technical system measuring residuals r_1, r_2 and r_3 from known variables, along with injected faults f_1, f_2 and f_3. Further, assume a *decision structure* can be constructed according to

	NF	f_1	f_2	f_3
r_1	0	X	0	X
r_2	0	0	X	X

where $X(i, j)$ denotes that a fault i is sensitive to residual j and NF denotes the no-fault case. In this example, this would mean that r_1 is sensitive to f_1 and f_3, while r_2 is sensitive to f_2 and f_3. Followed by Definition 2.1, f_2 is decoupled from r_1, f_1 is decoupled from r_2 while f_3 cannot be decoupled in any residual. Furthermore, by Definition 2.2 above these residuals are not sufficient to isolate all faults from each other. Since both r_1 and r_2 are sensitive to f_3 it is not possible to isolate f_1 and f_2 from f_3.

2.3 Principal Component Analysis

One method that has received tremendous attention regarding feature selection and dimension reduction is *Principle Component Analysis* (PCA). PCA preprocess the data before performing *Singular Value Decomposition* (SVD) in order to achieve a coordinate system that is determined by principle components which are orthogonal to each other and have strongest correlation with the measurements. With this, it is possible to go from a high-dimensional data set to a lower dimension that still explains the majority of the variance (often usual to explain 95% of the original data or choose a fixed number of components to use) [4].

Consider a large data set $\mathbf{X} \in \mathbb{R}^{p \times q}$

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x_1} & \mathbf{x_2} & \dots & \mathbf{x_q} \\ | & | & & | \end{bmatrix} \tag{2.1}$$

Where column $\mathbf{x_c} \in \mathbb{R}^p$ represents the measurements from residual q, and $p \gg q$. First step is to compute the mean row for all rows in $\mathbf{X}$ and then create a mean matrix with (2.2)

$$\bar{x}_j = \frac{1}{p} \sum_{i=1}^{p} X_{i,j} \tag{2.2}$$

$$\bar{\mathbf{X}} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \begin{bmatrix} \bar{x}_1 & \dots & \bar{x}_j \end{bmatrix} \tag{2.3}$$

The second step is to center the data in (2.1) with the mean in (2.3)

$$\mathbf{B} = \mathbf{X} - \bar{\mathbf{X}} \tag{2.4}$$

The third step is to compute the covariance matrix of $\mathbf{B}$ in (2.4)

$$\mathbf{C}_{BB} = \frac{1}{1-p} \mathbf{B}^\top \cdot \mathbf{B} = \begin{bmatrix} E[B_1 \cdot B_1] & E[B_1 \cdot B_2] & \dots & E[B_1 \cdot B_p] \\ E[B_2 \cdot B_1] & E[B_2 \cdot B_2] & \dots & E[B_2 \cdot B_p] \\ \vdots & \vdots & \ddots & \vdots \\ E[B_p \cdot B_1] & E[B_p \cdot B_2] & \dots & E[B_p \cdot B_p] \end{bmatrix} \tag{2.5}$$

Where $E[.]$ is the expected value of the scalar product of the two matrices. The covariance matrix measures how two matrices are related. For instance, if the covariance between two variables is positive, they move in the same direction and vice versa if negative [14].

Now, with the covariance matrix in (2.5), the fourth step is to compute the leading eigenvectors, which are related to the principle components of $\mathbf{X}$. Let $\mathbf{v}_1$ be the largest eigenvector to (2.5) that corresponds to the largest eigenvalue, λ_1. To get the first principle component u_1, $\mathbf{v}_1^\top \mathbf{C}_{BB} \mathbf{v}_1$ is computed, and repeats this for the other principle components. By using the property of *Eigenvalue Decomposition*, it is possible to split the $\mathbf{C}_{BB}$ matrix ($p \times p$) into a multiplication according to

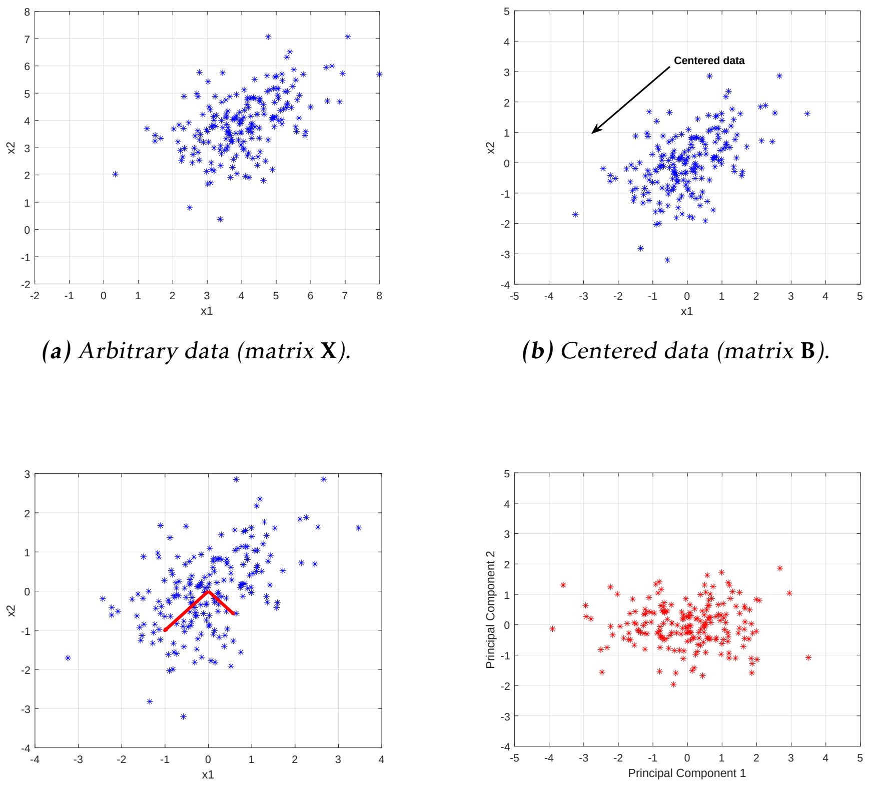
$$\mathbf{C}_{BB} = \mathbf{V}\mathbf{D}\mathbf{V}^{-1} =$$

$$= \begin{bmatrix} | & | & & | \\ \mathbf{v_1} & \mathbf{v_2} & \dots & \mathbf{v_p} \\ | & | & & | \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_p \end{bmatrix} \begin{bmatrix} | & | & & | \\ \mathbf{v_1} & \mathbf{v_2} & \dots & \mathbf{v_p} \\ | & | & & | \end{bmatrix}^{-1} \tag{2.6}$$

This leads to

$$\mathbf{C}_{BB} \cdot \mathbf{V} = \mathbf{V} \cdot \mathbf{D} \tag{2.7}$$

Where $\mathbf{V}$ is the eigenvectors, and $\mathbf{D}$ is the eigenvalues of $\mathbf{C}_{BB}$ [4]. A visualization of how it works is shown in Figure 2.5 below.



(a) Arbitrary data (matrix **X***).*

(b) Centered data (matrix **B***).*

(c) Eigenvectors of covariance matrix to **X***.*

(d) Data transformed to the new basis.

Figure 2.5: *Illustration of how PCA works.*

One thing to mention is that PCA allows the vectors in $\mathbf{V}$ creating the directions to be orthogonal to each other.

2.4 K-Nearest Neighbor

The *K-nearest neighbor* algorithm (KNN), has grown in the field of machine learning and pattern recognition due to its simplicity and effectiveness [1]. It works on the basis of two parameters:

- K - the number of nearest neighbors to a query sample, where $K \in \mathbb{N}$, $K > 0$.
- A distance metric - typically the euclidean distance is used.

Considering a data set of three classes with two numerical features, the goal is to classify which class label that new data belongs to by calculating the distance to the K nearest neighbors. Figure 2.6 below illustrates a case where new data from an unknown class is to be classified. With KNN, $K = 5$ nearest neighbors has been calculated to the new point.

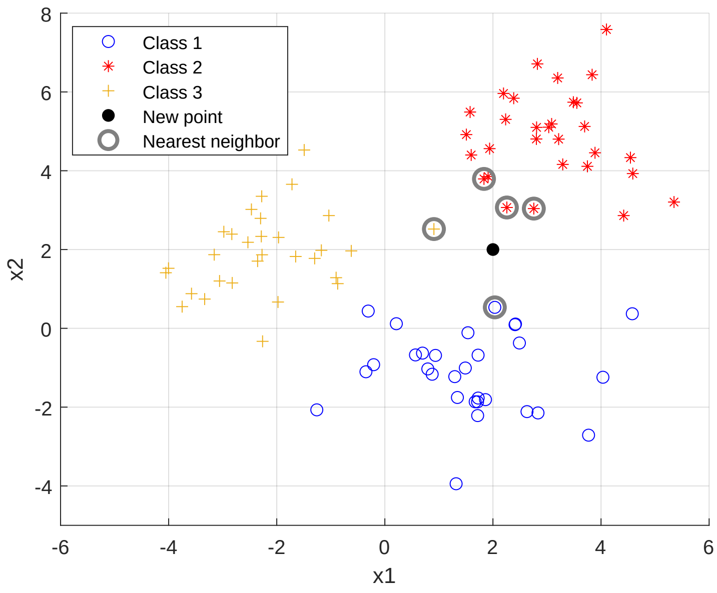


Figure 2.6: *Data from three known classes and the objective is to classify which of these the new data point belongs to. The encircled data points show the five nearest neighbors.*

Since three out of five nearest neighbors belonged to class 2 (red data), the algorithm would suggest that the new point belongs to class 2 (the red data). In this work, the euclidean distance metric has been used and follows the formula

$$\begin{aligned}\text{dist}(\mathbf{x}, \mathbf{y}) &= \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x_N - y_N)^2} \\ &= \sum_{i=1}^{n} (x_i - y_i)^2 \qquad (2.8)\end{aligned}$$

where $\mathbf{x}$ and $\mathbf{y}$ are two continuous vectors of both length n. Notice that equation (2.8) also can be used for higher dimensions. Other well-known distance metrics are: *Mahalanobis distance*, *Minkowski distance* and *City block distance* to mention a few.

The parameter K however is more intended to be tuned, unlike the distance measure, since its value is better suited for different applications where the data set might differ. In this thesis, the focus does not lie in finding an optimal value, but rather one that generates sufficient good results through trial and error.

2.5 Support Vector Machines

Support Vector Machine (SVM) is a powerful machine learning model, which can be used to perform linear or non-linear classification, regression but also outlier detection [10]. The fundamental objective of the SVM algorithm is to find a hyperplane that separates data points from different classes, which has the largest margins between them (may sometimes be refered to *maximum margin classifiers* in the literature) [1].

SVM is perhaps easiest to understand by an example. Consider a two-class classification problem (binary problem) where training data for respective class is available and comprises a class label for each observation

$$\text{Observations} = \begin{bmatrix} x_{1,1} & x_{1,2} \\ \vdots & \vdots \\ x_{n,1} & x_{n,2} \end{bmatrix}, \qquad \text{Class labels} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

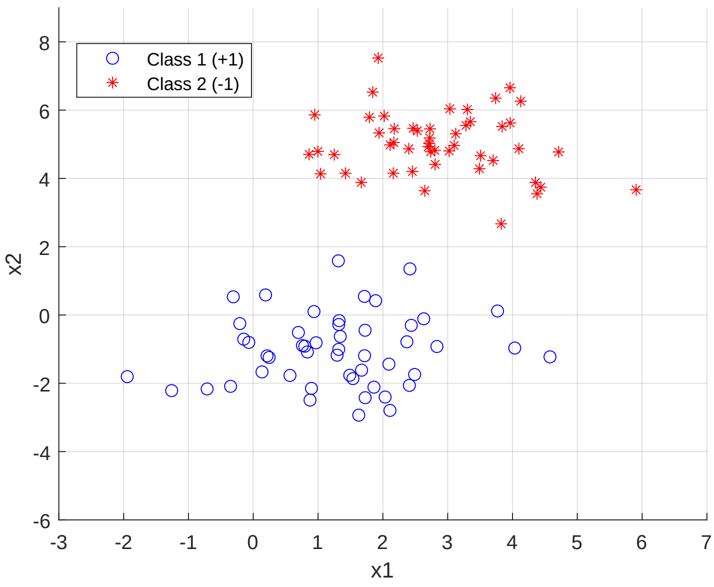


Figure 2.7: *An example of two-class problem which belongs to the easier classification setup for SVM since the classes can be linearly separated.*

where $y_i \in \{-1, +1\}$ which distinguish the two classes for each observation x_i.

Essentially, the SVM algorithm classifies new data by creating a classifier $h(x_i)$ that assigns $+1$ if $x_i \in S_{+1}$ and -1 if $x_i \in S_{-1}$, where S_* refers to the set of data belonging to class $y = *$. Although, one important aspect of how the algorithm defines the separable hyperplane depends on whether the data can be separated linearly or non-linearly. Lets assume the former, which may look like a problem illustrated in Figure 2.7 above.

Let S_{+1} and S_{-1} be the sets of data points belonging to the classes on each side of the hyperplane. Then, a hyperplane can be expressed as $g(x) = w^\intercal x + b$, where w is the normal vector and b is a constant bias term. This, of course leads to infinite many examples, therefore some constraints are necessary to find an optimal solution. This could be achieved by first creating two parallel hyperplanes, one for S_{+1} and one for S_{-1}, where their distance to the hyperplane is maximized. The region between the hyperplane for S_* and the separable hyperplane for the two classes is called margin and together they form a street, see Figure 2.8 below.

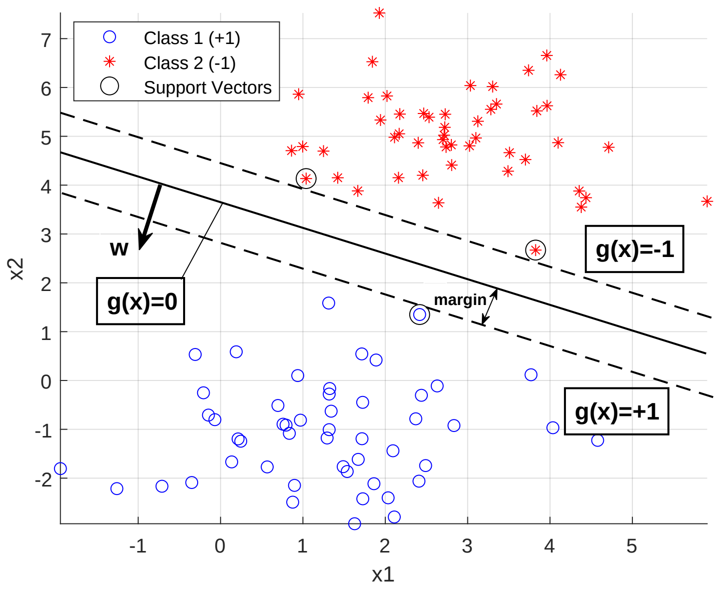


Figure 2.8: *An illustration of how the SVM hyperplane separates the two classes linearly. The dividing (solid) line between the two classes is characterized as $g(x) = 0$, while the dotted lines $g(x) = \pm 1$. The encircled data points closest to the hyperplane are called support vectors. They have high importance to the data sets since they mark the decision boundaries.*

The value of $g(x)$ depends on the magnitude of w, since the distance between the two dotted hyperplanes is $2/\|w\|$. Thus, the objective is to minimize this distance while still maintaining the data points from each class separated and to ensure that no data points are present in the margins between the hypeplanes.

The following optimzation problem can be formulated

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & y_i(w^\intercal x + b) \geq 1, \quad i = 1, \dots, n \end{aligned} \tag{2.9}$$

This is based on maximizing the minimum margin, i.e. the perpendicular distance from a point x_j in the training set. Now, (2.9) is an example of a quadratic programming problem that aims to minimizing a quadratic function subject to a set of linearly inequality constraints [1]. Solving this optimization problem leads to a classifier function

$$h(x) = sign(w^{*\intercal} x - b^*) \tag{2.10}$$

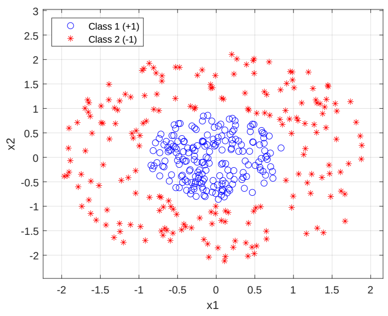
where w^* and b^* are the solution to the optimization problem in (2.9).

However, this approach tends to be sensitive for overfitting the decision boundaries since the constraints are strict. In order to loosen up the these conditions and allowing some training examples in S_{+1} and S_{-1} to appear in the margins, *Lagrange* multipliers are introduced along with slack variables [1].

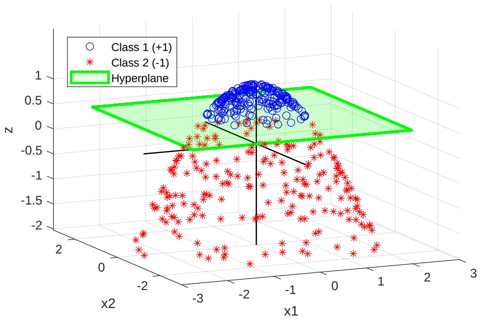
It is not always as simple to divide two classes linearly, which might be the case for many applications nowadays. Figure 2.9a shows an example where the two classes cannot be separated with a linear line, but requires a non-linear decision boundary. Fortunately, there is a way to solve this. It is based on mapping the data to a higher dimension, where it there can be separated linearly. This mapping, achieved with a kernel function, can be done differently, but one which has grown popular is radial-basis function (RBF) which has the following expression

$$K(x_i, x_j) = \exp\left(-\frac{\left\|x_i - x_j\right\|_2}{2\sigma^2}\right) \tag{2.11}$$

where σ is the width of the kernel and is a design parameter that influences how much of the training examples should be encapsulated around the decision boundary. Essentially, RBF is appropriate to use as kernel function when the data is non-linear. Figure 2.9b illustrates how a mapping with RBF is performed in order to create a decision boundary between the two classes.



*(**a**) This is a typical example where a linear decision boundary is insufficient to divide the two classes. Therefore a non-linear boundary is required.*



*(**b**) Plot that shows how the 2D data elevates to 3D with RBF and a hyperplane can separate the classes by then projecting back to 2D. This plot has been generated with [2].*

2.6 Error Correcting Output Codes

Error correcting output codes (ECOC) is a framework that decomposes a multi-class problem into a several binary problems, where each learner solves a sub-problem that uses a different class labelling [27]. The fundamental structure of the framework is to create a codeword for each class. The codewords are then arranged in a *coding matrix*, where the rows represent classes and the columns represent the learners (classifiers) [7], [23].

In this manner, an encoding matrix is obtained where each binary problem splits the classes into three possible partitions: $+1$, -1 or 0, which implies that the class is regarded as positive (target class), negative or zero (meaning the class is not considered in the current binary problem). When the algorithm attempts to predict an arbitrary test data, a decoding is made. Each data point in the test set generates a code that is compared to the base codewords for each class in the coding matrix. By using the *Hamming distance* metric the test data is assigned to the class with the closest codeword [16], [27].

Working with ECOC, there are different approaches when designing the coding matrix. Two common methods are *one-vs-one* (OVO) and *one-vs-all* (OVA). The former aims at constructing a coding matrix where a pair of classes are available and the rest ignored, see Table 2.1. The number of binary learners in this case is $K(K-1)/2$, where K is the number of classes.

Table 2.1: *Decomposition of multi-class problem according to OVO.*

	Learner 1	Learner 1	Learner 3
Class 1	1	1	0
Class 2	-1	0	1
Class 3	0	-1	-1

On the other hand, OVA lets each binary learner assign one class as positive and the remaining as negative, thus taking all others into consideration when solving each binary problem, see Table 2.2. The number of binary learners in this case is K.

Table 2.2: *Decomposition of multi-class problem according to OVA.*

	Learner 1	Learner 1	Learner 3
Class 1	1	-1	-1
Class 2	-1	1	-1
Class 3	-1	-1	1

The decomposed binary classification problems can be solved with various machine learning algorithms. In this work SVM is chosen to be studied further.

3 Method

An interesting aspect of working with model-based fault diagnostics is how different faults relate to each other. Common methods to analyse their dependencies are by fault isolation and decoupling. These methods are highly dependent on explicitly and accurate derived models that tries to describe the system in different operation points, and the idea behind these is to identify any deviations from the nominal case in order to identify a fault. This can be time-consuming and for large, complex systems it can even be infeasible.

Working with data-driven fault diagnostics, one instead turns to rely heavenly on available training data that represents the system working on various operating points. Assuming training data is at disposal from different fault scenarios, the methodology in this chapter attempts to investigate the possibility to decouple a fault in a data-driven fashion and then try to classify new data to draw a final diagnosis. Lastly, a complete diagnostic framework is presented on how it could be handled systematically along with a proposed method to estimate the fault size of a diagnosed fault.

3.1 Decoupling of fault using PCA

In model-based diagnosis, the isolability of faults is studied by decoupling faults in different residuals. In this thesis however, a different fault-decoupling approach is investigated. Rather than working with models, a data-driven method is used which is described here.

Consider a set of four classes and each class has its own feature vectors, where the data points are spread out over the feature space $\in \mathbb{R}^3$, see Figure 3.1. Let $\mathbf{X}_i = \left[\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\right]$, $\mathbf{X} \in \mathbb{R}^{n\times 3}$, be the feature space for Class i, $i = 1, \ldots, 4$.

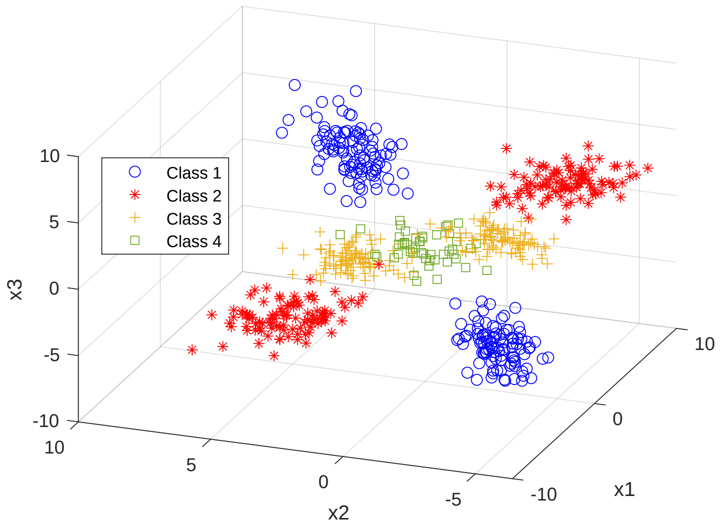


Figure 3.1: An illustration of how data from different classes look like. Data from Class $1, 2, 3$ *represents coming from different faults, while Class* 4 *is supposed to represent the* NF*-case (No Fault) which is centered at the origin with no specific direction.*

By performing PCA, it is possible to find the *fault vector*[1] for each class. Essentially, crucial information can be extracted from the fault vector, such as how much percentage it solely explains of the total variance in the data, but also its orientation. The former can be thought as a measurement to the data itself on how important it is in comparison to the other principal components. The greater percentage the fault vector can explain, the more data will be tighter projected around the origin to its subspace, see Figure 3.2. The latter is a measurement to compare the angle between the fault vector and the base-features to study how easy it is to isolate and decouple a feature, additionally to compare the angle between fault vectors from different classes. Therefore, the *cosine similarity* (3.1) is a simple yet effective method to conduct these comparisons [13].

$$cos(\theta_{i,j}) = \frac{\mathbf{F}_i \cdot \mathbf{F}_j}{\|\mathbf{F}_i\|_2 \left\|\mathbf{F}_j\right\|_2} = \frac{\sum_{m=1}^{n} F_{i,m} F_{j,m}}{\sqrt{\sum_{m=1}^{n} F_{i,m}^2}\sqrt{\sum_{m=1}^{n} F_{j,m}^2}} \tag{3.1}$$

One prerequisite to use (3.1) is that the two vectors have equal length and are non-zero. Two special cases are of interest to investigate [2]

[1]A fault vector F_i is the line along which data from class i lies on. A geometric interpretation is the direction (from the origin) in which data is headed.

[2]Note, these expressions also applies when comparing the fault vector to the base-features, i.e. the residuals.

- If $\mathbf{F}_i \perp \mathbf{F}_j \implies cos(\theta_{i,j}) = 0$
  This implies that the two vectors are (ideally) unrelated.

- If $\mathbf{F}_i \parallel \mathbf{F}_j \implies cos(\theta_{i,j}) = 1$
  In this case, the vectors have maximum relation. This would mean that the subspaces to $\mathbf{F}_i$ and $\mathbf{F}_j$ are parallel, meaning that no uniquely information can be found in one of the subspaces but not in the other.

One thing to point out from this similarity measure is that the angle determines how easy it is to isolate a fault from another, but also to tell if a residual is decoupled from a specific fault. Decoupling a residual would for instance reduce the dimension on the residual data, i.e. work as a feature selection, thus be beneficial from a time perspective.

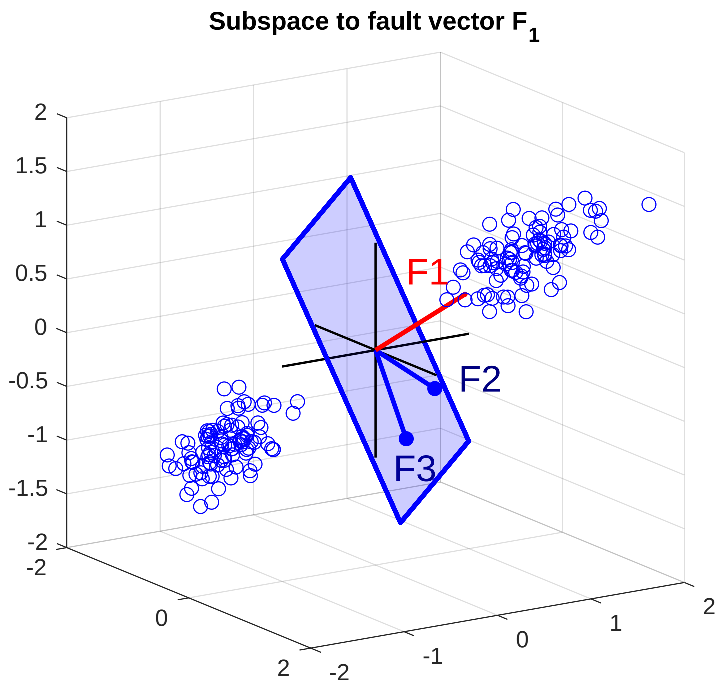


__Figure 3.2:__ A visualization of PCA performed on Class 1, where the subspace that is spanned by the principal components F2 and F3 is illustrated. This plot has been generated with [2].

The next step is to decouple one class at the time by projecting all data onto its corresponding subspace, thus resulting in a subspace for each class where the class data is decoupled. An illustration of how it might look like can be seen in Figure 3.3.

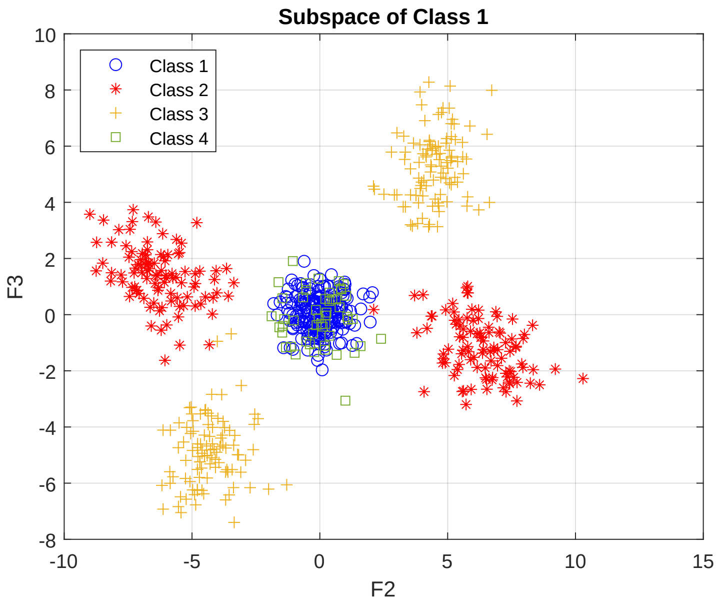


Figure 3.3: *The subspace of Class* 1 *is shown, where all data from Figure 3.1 has been projected onto. The blue data points belonging to Class* 1 *gets projected around the origin. Notable, the data from Class* 4 *(the NF-case) also gets projected around the origin, which is common to all subspaces.*

At this point, a classification procedure is needed to both identify which class new data comes from and to determine if it comes from an unknown class. Although, one thing to bear in mind is that the classification procedure may vary depending on what extent one wants to take it, or how much training data that is available for that matter. One way, that follows a simple strategy, to determine where new data originates from is to train a binary classifier for each subspace which separates the decoupled class from the rest, i.e. OVA technique (see Figure 3.4 below). In this manner, each subspace would give an output of how likely that new data belongs to the decoupled class and if no subspace provides a sufficiently high confidence it is deemed as an unknown class. Furthermore, another way to analyse how much new information that can be found between different subspaces, in contrast to measuring the angle between each fault vector, is to analyse the correlation of the binary outputs produced by these classifiers.

In [25], the authors suggested a method to calculate a correlation matrix based on the binary classifiers output. Assume each binary classifier produces an outcome $\in \{-1, +1\}$, where $+1$ indicates that the query sample is classified as the target class (in this example the decoupled fault) and -1 means that the query sample is an outlier, i.e. belongs to any of the counterexamples to the decoupled fault. Let O_i and O_j denote the outcome from classifiers i and j, respectively. Then, the correlation between two classifiers is computed as

$$A_{i,j} = \frac{1}{n}\left| \sum_{\forall\, x_q \in \mathbf{X}_{Te}} O_i(x_q) \cdot O_j(x_q) \right| \tag{3.2}$$

where $\mathbf{X}_{Te}$ is the test data and n is the number of observations in $\mathbf{X}_{Te}$ (note the scalar product between the outputs in (3.2)). With this measurement, the generated matrix $\mathbf{A}$ would point out how well two classifiers relate. For instance, if the outputs for all samples in $\mathbf{X}_{Te}$ coincide, e.g. produces only $+1$ or -1, then their correlation is one. On the other side, if they always disagree the correlation is zero [25].

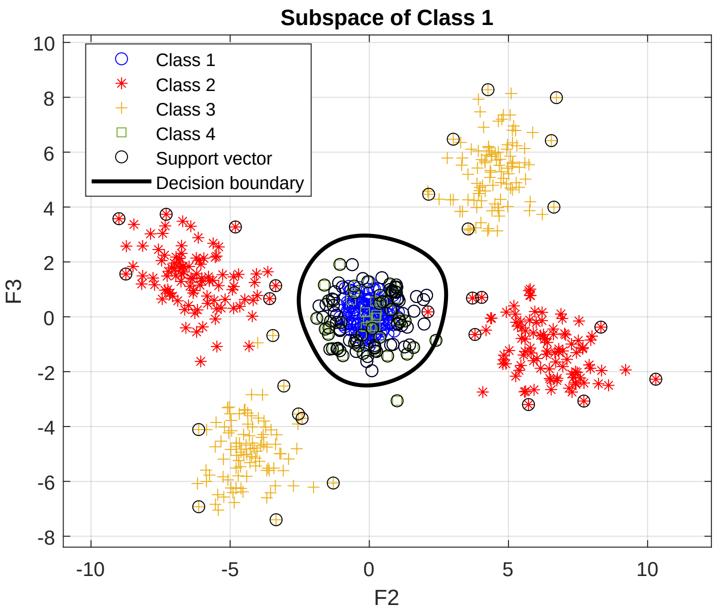


Figure 3.4: *Illustration of a binary decomposition with OVA technique. This is an example of a binary SVM with a RBF kernel function that aims at capturing the property of the decoupled class (blue data) while taking counterexamples from the other classes into account when determnining the decision boundary.*

Figure 3.5 below is another example of how classification on a subspace could be performed. It is based on decomposing the multi-class problem with one-class classifier (OCC) instead, that aims at capture the unique property of a class with no counterexamples at disposal.

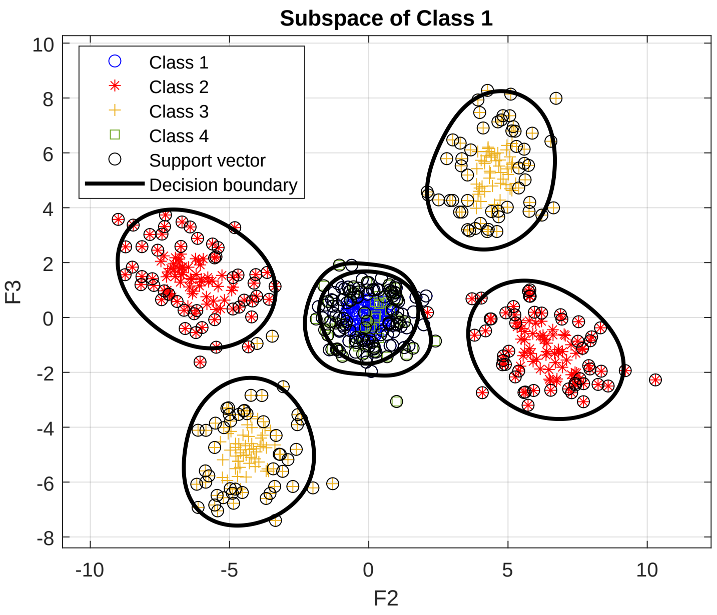


Figure 3.5: *This classification is based on training one-class support vector machines (OCSVM) models for each class on a local subspace, with RBF as kernel function.*

Another way to construct this classification problem would be to train a multi-class classifier (MCC) for each subspace that tries to distinguish all the classes apart. Nevertheless, there are many different methods to use, but in this study the aim is to investigate how an information fusion with these classifiers could be performed. This example is intended to show to how decoupling of a fault class could be handled but also the classification possibilities that comes along with it.

3.2 Data Processing by Reshaping

One useful benefit when working with time-series data, such as measured signals from sensors, is the possibility to take multiple observations when training models into consideration. Assuming a residual data $\mathbf{X}$ consist of c columns where each one represents a residual r_i of length n

$$\mathbf{X} = \begin{bmatrix} | & & | \\ \mathbf{r_1} & \dots & \mathbf{r_c} \\ | & & | \end{bmatrix} \tag{3.3}$$

Then a partitioning of each residual can be made by dividing it into N batches, where each batch gets inserted as a new column in $\mathbf{X}$ yielding $\mathbf{X}_{new}$. Figure 3.6 illustrates how it is carried through.

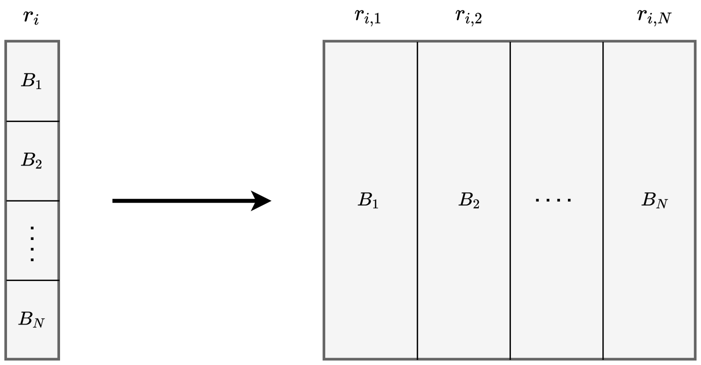


Figure 3.6: *Illustration of how an arbitrary residual i in the residual data reshapes, where B indicates a batch of data in r_i.*

The generated $\mathbf{X}_{new}$ gets stretched into $N \cdot c$ columns, while the number of rows n shrinks to be adjusted accordingly. One thing to keep in mind when working with such preprocess technique is how the data varies over time, whether there are consistent trends or has the characteristics of a *random walk*.

3.3 Classifier Selection

A classifier selection from a pool of classifiers can be seen as a filtering process, where the goal is to eliminate non-competent classifiers to the new test data. The classifier selection method used in this thesis is named *Threshold-Based neighborhood pruning*, proposed by [17]. Although the authors based this method on a set of OCC, it has shown promising results when using for binary and multi-class problems. Especially in a multi-class problem, there is a need for decomposition in order to reduce the complexity, but also the classification performance. The full algorithm is shown in Algorithm 1 below. The idea behind this selection process is that noisy data or outliers from an arbitrary class should not have a strong influence on the local (competence) region to the test data. This type of dynamic classifier selection is based on measuring the distance from known data to test data with KNN. With a threshold parameter J, a filtering is made where classes that does not occur above it are considered as non-competent to the local region to the test data. Likewise in the mentioned article, the parameter is set to $J = 0.1$, i.e. classes occurring less than 10% are eliminated from the next step in the process. Another parameter required in the algorithm is the number nearest neighbor K, which is set to $3 \cdot N$, where N is the number of base learners.

There are numerous alternatives when it comes to different measurement techniques, however the euclidean distance is used as default in this study. Other algorithms using KNN that have gotten established are mentioned in [3]. Moreover, there are other types of measurement options rather than solely looking at distance when it comes to selecting competent classifiers to the test data. For

Algorithm 1 Threshold-Based neighborhood pruning

Input: Pool of Classifiers (POC), Partitioned Data (PD), Test Data (TeD), K, J
Output: Local Classifiers (LC), Occurrences & Proportion of each fault class (OoF)

```
 1: for each training partition $pt_i \in PD$ do
 2:     $X_{Tr} \leftarrow pt_i$ (residual data)
 3:     $Y_{Tr} \leftarrow pt_i$ (fault class)
 4: end for
 5: $Counter \leftarrow 0$
 6: for each sample $t \in TeD$ do
 7:     Compute $K$ nearest neighbors to $X_{Tr}$          ▷ e.g. with euclidean distance
 8:     for each $k \in K$ do
 9:         Determine the class label to the $k$:th neighbor
10:         $Counter \leftarrow$ occurrence of class label from $k$
11:     end for
12: end for
13: for each class label $\omega_j$ do
14:     $OoF.occurrences \leftarrow$ total occurrences of $\omega_j$ from $Counter$
15:     $OoF.proportion \leftarrow$ the proportion nearest neighbor $\omega_j$ has to $TeD$
16:     if $OoF.proportion$ of class label $\omega_j \geq J$ then
17:         Add classifier for $\omega_j$ from $PoC$ to $LC$          ▷ Threshold pruning
18:     end if
19: end for
```

instance, the *Kullback-Leibler divergence* is another way of measuring the competence of a set of classifiers described in [29], [6].

3.4 Information Fusion

Aggregating decision outputs from multiple classifies can be done in various ways. In this study however, the method used is limited to the one proposed in article [5], along with some modifications to adjust it for this type of problem. The proposed method by the authors performs a combination of classifier fusion and classifier selection, which makes it a hybrid approach. Rather than using dynamic classifier selection by local accuracy (DCS-LA) proposed by [30] to choose the (single) most local competent classifier, the authors instead used a similar approach to determine the best weights for the most competent classifiers to fuse decision outputs.

In the following section, firstly the method is described when using MCC and secondly for the case when using binary classifiers.

3.4.1 Local Classifier Weigthing by Quadratic Programming - Multi-Class Classifier

To begin with, the authors in [5] defines some mathematical terms. Assume $\Psi = \{\Psi_1, \ldots, \Psi_L\}$ is a set of L locally expert classifiers[3] and $\Omega = \{\omega_1, \ldots, \omega_M\}$ is a set of M class labels. Each classifier Ψ_i outputs a hypothesis vector

$$\boldsymbol{\psi}_i(\boldsymbol{x}_q) = \left[\psi_{i,1}(\boldsymbol{x}_q), \ldots, \psi_{i,M}(\boldsymbol{x}_q)\right]^\mathsf{T} \tag{3.4}$$

where $\psi_{i,j}(\boldsymbol{x}_q)$ denotes the support from classifier i to class label j. Importantly, depending on type of classifier[4], this support can be interpret as an estimate of posterior probability $p(\omega_j|\boldsymbol{x}_q)$. It follows that

$$\psi_{i,j}(\boldsymbol{x}_q) \in [0,1] \quad \text{and} \quad \sum_{j=1}^{M} \psi_{i,j}(\boldsymbol{x}_q) = 1$$

With this, the aim is to compute weights for each classifier, thus enabling to combine the classifier outputs in order to label a query sample $\boldsymbol{x}_q$. It can be written

[3]Locally expert classifier refers to being close to the region of test data, implying that the classifier can contribute in the decision making.

[4]Not all classification algorithms share the same type of output. For instance classifiers based on SVM offers a *sign* on the prediction score that determines if a query sample can be explained by a class or not.

as

$$\boldsymbol{\alpha}(\boldsymbol{x}_q) = \left[\alpha_1, \dots \alpha_L\right]^\mathsf{T}, \quad \text{where}$$
$$\alpha_i \geq 0 \quad \text{and} \quad \sum_{i=1}^{L} \alpha_i = 1 \tag{3.5}$$

Finally, once the weights have been established the final support for each class label can be computed as the weighted sum of the hypothesis for each classifier in Ψ

$$p(\omega_j|\boldsymbol{x}_q) = \sum_{i=1}^{L} \alpha_i \psi_{i,j}(\boldsymbol{x}_q), \quad j = 1, \dots, M \tag{3.6}$$

and the query is assigned to the class label ω_j that generates the highest estimated posterior probability

$$\boldsymbol{x}_q \in \omega_m, \quad p(\omega_m|\boldsymbol{x}_q) = \max_{j=1,\dots,M} \{p(\omega_j|\boldsymbol{x}_q)\}$$

As mentioned, this type of problem requires the classifiers to produce a probability regarding how likely that a new sample belongs to a certain class. Using SVM as base learners, it is possible to map the produced scores with a sigmoid function [22], which produces estimates of posterior probability.

Determining the weights

The weight estimation is based on minimizing the (local) classification error, which defines the classifier accuracy. The local regions to $\boldsymbol{x}_q$ are defined by calculating the K-nearest neighbors (as in DCS-LA).

Let $\mathbf{t}(\boldsymbol{x}_k) = \left[t_1, \dots, t_M\right]^\mathsf{T}$ be a class index vector, which jth component is 1 and all other 0, if the k:th nearest neighbor $\boldsymbol{x}_k$ comes from class ω_j. For instance, if $\boldsymbol{x}_k$ comes from ω_2, then $\mathbf{t}(\boldsymbol{x}_k) = \left[0, 1, 0, \dots, 0\right]^\mathsf{T}$.

The weight estimation problem can then be formulated as

$$\min_{\alpha(\boldsymbol{x}_q)} \quad \sum_{k=1}^{K} \left\| \mathbf{t}(\boldsymbol{x}_k) - \sum_{i=1}^{L} \alpha_i \boldsymbol{\psi}_i(\boldsymbol{x}_k) \right\|^2$$
$$\text{s.t.} \quad \alpha_i \geq 0 \quad \text{and} \quad \sum_{i=1}^{L} \alpha_i = 1 \tag{3.7}$$

Using the constraints on $\boldsymbol{\alpha}$ in (3.5), the error corresponding to a neighbor $\boldsymbol{x}_k$ can then be written as

$$\epsilon(\boldsymbol{x}_k) = \left\| \sum_{i=1}^{L} \alpha_i \left[\boldsymbol{\psi}_i(\boldsymbol{x}_k) - \mathbf{t}(\boldsymbol{x}_k)\right] \right\| \tag{3.8}$$

Equation (3.8) can be rewritten as $\boldsymbol{\alpha}(\boldsymbol{x}_q)^\intercal \mathbf{A}^{(k)} \boldsymbol{\alpha}(\boldsymbol{x}_q)$, where

$$\mathbf{A}^{(k)} = \left(A_{i,j}^{(k)}\right)_{i,j=1,\ldots,L} = \left[\boldsymbol{\psi}_i(\boldsymbol{x}_k) - \mathbf{t}(\boldsymbol{x}_k)\right]\left[\boldsymbol{\psi}_j(\boldsymbol{x}_k) - \mathbf{t}(\boldsymbol{x}_k)\right] \tag{3.9}$$

Note that there is a dot product between the vectors in (3.9). Now, by letting $\mathbf{A} = \sum_{k=1}^{K} \mathbf{A}^{(k)}$, a weight estimation problem can be written as

$$\begin{aligned} \min_{\alpha(\boldsymbol{x}_q)} \quad & \boldsymbol{\alpha}(\boldsymbol{x}_q)^\intercal \mathbf{A} \boldsymbol{\alpha}(\boldsymbol{x}_q) \\ \text{s.t.} \quad & \alpha_i \geq 0 \quad \text{and} \quad \sum_{i=1}^{L} \alpha_i = 1 \end{aligned} \tag{3.10}$$

The expression in (3.10) is a quadratic problem. Since $\mathbf{A}$ is a symmetric positive semi-definite matrix the objective function is convex, which means that a global minimum exist [5].

The authors in [5] also expands the objective function with a term regarding the confidence of the classifiers decision. This additional term is aimed to guide the classifiers in the decision making whether they agree on the label belonging to $\boldsymbol{x}_q$. Assuming classifier Ψ_i assigns $\boldsymbol{x}_q$ to class ω_j, then the corresponding confidence of the classifier is estimated as

$$\beta_i = \frac{N_{TP}}{N_{TP} + N_{FP}} \tag{3.11}$$

where N_{TP} (true positive) is the number of nearest neighbors correctly classified as ω_j and N_{FP} (false positive) is the number of nearest neighbors wrongly classified as ω_j. The expression in (3.11) here is a shortened version of the one in [5]. Adding this measurement to (3.10) yields

$$\begin{aligned} \min_{\alpha(\boldsymbol{x}_q)} \quad & \boldsymbol{\alpha}(\boldsymbol{x}_q)^\intercal \mathbf{A} \boldsymbol{\alpha}(\boldsymbol{x}_q) - \gamma \boldsymbol{\beta}^\intercal \boldsymbol{\alpha}(\boldsymbol{x}_q) \\ \text{s.t.} \quad & \alpha_i \geq 0 \quad \text{and} \quad \sum_{i=1}^{L} \alpha_i = 1 \end{aligned} \tag{3.12}$$

where $\boldsymbol{\beta} = \left[\beta_1, \ldots, \beta_L\right]^\intercal$ and γ is a regularization parameter which regulates the tradeoff between local classification accuracy and classifier confidence.

To solve the optimization problem (3.12), CVX is used, a package for specifying and solving convex programs [11], [12].

3.4.2 Local Classifier Weigthing by Quadratic Programming - Binary Classifier

A binary classifier, unlike MCC, can only produce an output of two classes where it in certain sense comes down to true or false statement. The binary classifiers in this study follows the construction according to Section 3.1, where the SVM algorithm and OVA technique is used. Now, this type of binary problem might differ from the typical ones, since each binary classifier is trained on its own decoupled fault as target class while the other class consists of data from the remaining faults. Therefore, some adjustments are required.

The hypothesis vector in (3.4) becomes instead

$$\boldsymbol{\psi}_i(\boldsymbol{x}_q) = \left[\psi_{i,T}(\boldsymbol{x}_q), \psi_{i,O}(\boldsymbol{x}_q)\right]^\intercal \tag{3.13}$$

where indices T and O correspond to target respective outlier class. Mapping the decision outputs with sigmoid function as mentioned above, these outputs represents posterior probabilities. Essentially, $\psi_{i,O}(\boldsymbol{x}_q)$ is the probability that the query sample belongs to any of the other faults in the current subspace. Therefore, as a simplification it is assumed that the probability is equally distributed over all the other fault classes. It follows that the probability a query sample belongs to any of these classes is $\frac{\psi_{i,O}(\boldsymbol{x}_q)}{M-1}$. With this the hypothesis vector in (3.13) can now have the same appearance as (3.4).

Determining the weights is done similarly and (3.10) can be formulated as in the case for MCC. However, the expansion of the objective function using confidence measure with equation (3.11) is not appropriate to use in this setup. A different approach for evaluating classifiers performance and taking it into account in the objective function is needed, in hope it will boost the decision making.

One measure is the *mean-squared error* (MSE). It measures the average squared difference between two vectors, or in other words the average error. In order to adapt it in this problem, let the difference be calculated by the query sample and every support vector that belongs to classifier i

$$\mathbf{X}_{SV} = \begin{bmatrix} | & | & & | \\ s.v_1 & s.v_2 & \dots & s.v_n \\ | & | & & | \end{bmatrix}^\intercal, \qquad \mathbf{X}_q = \begin{bmatrix} | & | & & | \\ \boldsymbol{x}_q & \boldsymbol{x}_q & \dots & \boldsymbol{x}_q \\ | & | & & | \end{bmatrix}^\intercal$$

$$MSE_i = \frac{1}{n}\sum_{j=1}^{n}(\mathbf{X}_{q,j} - \mathbf{X}_{SV,j})^2 \tag{3.14}$$

where n is amount of support vectors in classifier i. A large value on MSE$_i$ would point out that the query sample $\boldsymbol{x}_q$ deviates much to the averaged support vector belonging to classifier i, while a small value would indicate that $\boldsymbol{x}_q$ is on average close to the decision boundary. Repeating this for each classifier, the optimization problem in (3.10) can be expanded to

$$\begin{aligned} \min_{\alpha(\boldsymbol{x}_q)} \quad & \boldsymbol{\alpha}(\boldsymbol{x}_q)^\intercal \mathbf{A} \boldsymbol{\alpha}(\boldsymbol{x}_q) + \gamma \boldsymbol{\eta}^\intercal \boldsymbol{\alpha}(\boldsymbol{x}_q) \\ \text{s.t.} \quad & \alpha_i \geq 0 \quad \text{and} \quad \sum_{i=1}^{L} \alpha_i = 1 \end{aligned} \tag{3.15}$$

where $\eta = \left[MSE_1, \dots, MSE_L\right]^{\mathsf{T}}$ and γ again being a regularization parameter. The optimization problem (3.15) may aswell be solved with CVX as mentioned previously.

3.5 Fault Size Estimation

In a scenario where test data has been classified as one of the known fault classes available in the training data, it could be of interest to estimate the corresponding fault size since it could point out the severity. This would for instance assist an operator when deciding the level of urgency if a component needs to be repaired or replaced.

One way to do this is to make use that the residuals values increases or decreases for different fault sizes. Assuming they change linearly[5], the MSE could be used as a measurement to determine the deviation from an ideal nominal case. Let MSE be calculated for data from each fault size with the origin as reference term. Then a mapping could be made, i.e. for each fault size corresponding to fault i there exist MSE_i, which allows for a linear model to be fitted. Using a method such as *least squares* the slope of the line that best fits the data points, which passes through the origin for natural reasons, could be calculated.

Consider a linear model on the form $y_{FS} = \kappa x$. The least squares solution lies in finding the κ that minimizes

$$\sum_i (\kappa x_i - y_{FS,i})^2 \tag{3.16}$$

where x_i is the MSE for an arbitrary fault size and $y_{FS,i}$ is the corresponding fault size estimation.

3.6 Complete Diagnostic Framework

The complete diagnostic framework is presented in this section and can be divided into three parts: preparation, classification and evaluation. An overview of the framework can be seen in Figure 3.7.

[5]In Chapter 4, expression (4.2) explains that sensor faults changes linearly for different fault sizes, thus it is considered as an appropriate assumption that the residuals also does.

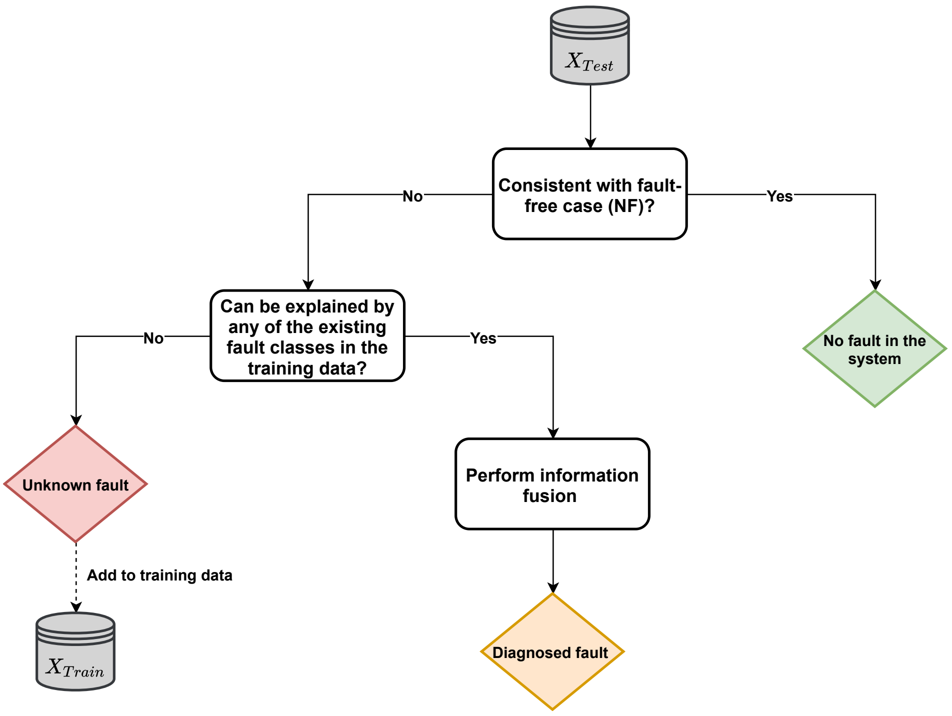

X_{Test}
Consistent with fault-free case (NF)?
No
Yes
Can be explained by any of the existing fault classes in the training data?
No fault in the system
No
Yes
Unknown fault
Perform information fusion
Add to training data
X_{Train}
Diagnosed fault

Figure 3.7: *Presentation of how a complete diagnostic framework could look like, which is what this study strive. This procedure covers the three most important parts in fault diagnostics; identifying a fault-free system (preventing false detecting), identifying unknown data (might be a new fault class or a new realization of an existing fault class) and lastly to draw a final diagnosis statement.*

3.6.1 Preparation

In order to create a robust diagnostic framework, a good preparation is necessary. Firstly, it consist of preprocessing by partitioning the available residual data into two parts; training and validation. Secondly, with the training subset PCA is performed which will generate the fault vector and the transformation matrix that projects data onto the fault in question's subspace. Essentially, the transformation matrix is used to project training data, validation data and any new incoming test data onto each subspace. Thirdly, classifiers are trained in each subspace. To identify whether new incoming test data is recognized by any of the existing training examples, binary SVM are trained where the target class is the decoupled fault and the other class is data from the other faults. Furthermore, a MCC for each subspace is also trained in order to perform an information fusion if necessary. To ensure that the classifiers meet satisfied degree of performance, they are validated.

3.6.2 Classification

In the classification stage is where test data is loaded and the proposed diagnostic framework is tested. The procedure follows according to Figure 3.7. The first thing to check is whether the test data is consistent with fault-free case, since if it is true then there is no need to diagnose a fault hypothesis. However, if the statement is false then the next step is to determine if it comes from an unknown fault. It is not until the diagnostic framework is confident that the test data must come from any of the existing fault types that an information fusion is performed. Once a fault has been diagnosed, a fault size estimation is done to determine the severity of the fault.

3.6.3 Evaluation

The evaluation stage involves investigating the classification from two aspects. Firstly, it is used to study properties of the data sets, such as the cosine similarity between fault vectors. It is meant to investigate how much similarity there is between different faults, in order to tell how difficult it is to carry through the method and what results to expect. Secondly, to compare the diagnostic framework with a standard classification algorithm retrieved from state-of-the-art, namely a *Random Forest classifier* [9].

4 Results

The diagnostic framework is tested and evaluated by using experimental data collected from an engine test bench at Linköping University, Division of Vehicle Systems at the Department of Electrical Engineering. The engine is a commercial, turbo charged, four cylinder, ICE from Volvo Cars. A schematic view of the engine can be seen in Figure 4.1, where y denote sensor measurements and u denote actuator signals.

4.1 Data Collection

This Master's thesis is based on residual data from previous work where the data has been generated with neural network [15]. Measured data has been collected from various operating points along with different (single) faults and fault magnitudes. The residual data has been generated by calculating the difference between measured values and predicted values according to

$$\mathbf{r} = \mathbf{y} - \hat{\mathbf{y}} = \begin{bmatrix} | & & | \\ \mathbf{r}_1 & \dots & \mathbf{r}_9 \\ | & & | \end{bmatrix} \tag{4.1}$$

Notable, the residual data consist of nine residuals brought from the mentioned article. In Table 4.1 a list of the fault classes are presented, where each fault class represents a specific area in the engine and the fault-free class is the nominal system operation.

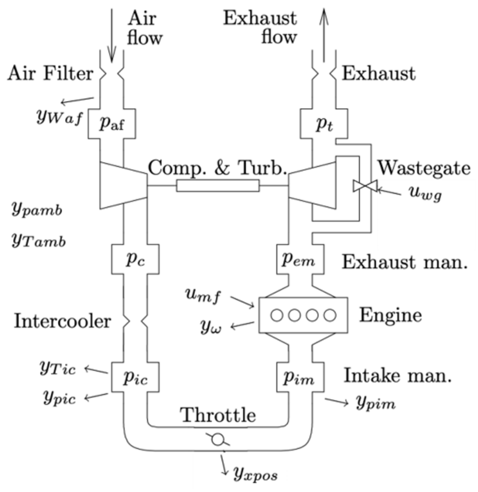


Figure 4.1: *Schematic of the model of the air flow through the model. This figure is used with permission from [8].*

Table 4.1: *The considered fault cases that are studied in this master thesis.*

Fault Class	Description
NF	Fault-free
f_{pim}	Fault in sensor measuring pressure in intake manifold
f_{pic}	Fault in sensor measuring pressure after compressor
f_{waf}	Fault in sensor measuring air mass flow after air filter
f_{iml}	Leakage after throttle
f_{acl}	Leakage after compressor
f_{bcl}	Leakage before compressor
f_{af}	Clogging in air filter

The sensor faults listed in Table 4.1 are injected by multiplying the measured variable z_i in each sensor y_i by a factor θ such that the output is given as

$$y_i = (1 + \theta) \cdot z_i \tag{4.2}$$

where $\theta = 0$ corresponds to the nominal case. The leakage faults are presented with a unit of length, e.g. $4\,mm$, which corresponds to a leakage with hole diameter of $4\,mm$. In Table 4.2 a list of the available training data with different fault sizes used to train classifiers is presented.

Table 4.2: *The fault classes and known magnitudes that are represented in the training data.*

Fault Class	**Fault magnitudes**
NF	-
f_{pim}	−20%, −15%, −10%, −5%, +5%, +10%, +15%
f_{pic}	−20%, −15%, −10%, −5%, +5%, +10%, +15%
f_{waf}	−20%, −15%, −10%, −5%, +5%, +10%, +15%, +20%
f_{iml}	4 mm, 6 mm and two unknown fault sizes

To get a feeling on what impact the fault size has on the residuals, Figure 4.2 below shows how the deviation of residual three, i.e. $\mathbf{r}_3$, changes from the nominal case when changing the fault size.

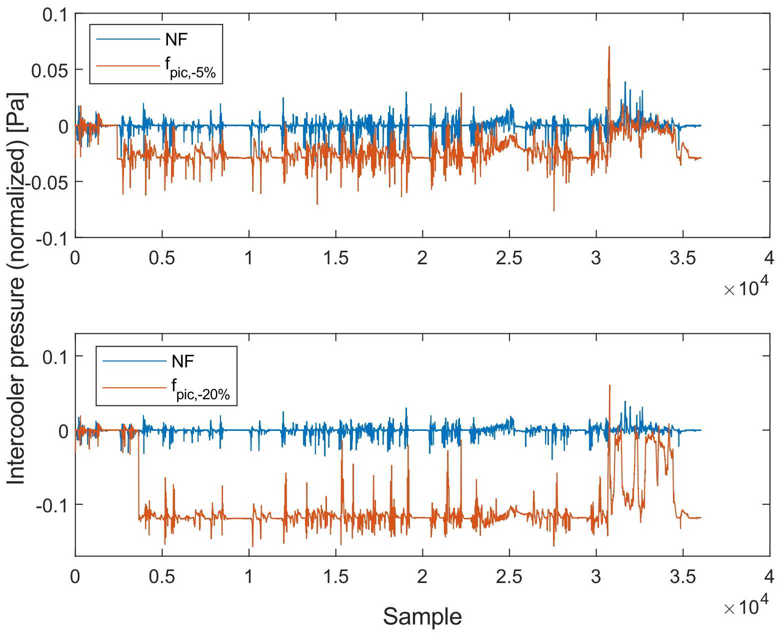


Figure 4.2: *Plot that shows the intercooler pressure from NF and f_{pic} when incrementing the fault size −5% → −20%. This indicates that smaller fault sizes are closer to the nominal case which makes them more difficult to distinguish. In general, this applies for all the listed fault classes in Table 4.2.*

4.2 Preprocessing

The first thing done in the proposed method is to preprocess the data. It starts by gathering residual data from all fault classes with all available fault sizes in matrices, yielding $\mathbf{X}_{fpim}, \mathbf{X}_{fpic}, \mathbf{X}_{fwaf}, \mathbf{X}_{fiml}$ and $\mathbf{X}_{NF}$. Then a holdout partition, using Matlab's in-built function *cvpartition*, is performed dividing the matrices into two randomized parts, one for training and the other for validation $(70 + 30)\%$. With the training part, PCA is applied which finds the fault vectors (see Table 4.3) and the projection matrix to the corresponding subspace.

Table 4.3: *The fault vectors with corresponding explained total variance they captures. Higher percentage implies that more data moves in the same direction which as an indirect result will project that data tighter around the origin on its subspace.*

Fault vector	Explained total variance [%]
$F_{1,fpim}$	63.1663
$F_{1,fpic}$	77.8520
$F_{1,fwaf}$	72.4052
$F_{1,fiml}$	68.5249

With cosine similarity (3.1), the angle between the data in fault vector's direction and the base residuals for fault i can be calculated to investigate which residuals that are easy or hard to decouple, see Table 4.4 below.

***Table 4.4:** The angle between the fault vectors with each base-residual. Keeping in mind that angles close to 0° means that a specific residual is hard to decouple from the fault and vice versa for angles close to 90°.*

$F_{1,fpim}$	Degree (°)	$F_{1,fpic}$	Degree (°)
r_1	125.3720	r_1	13.8245
r_2	10.2984	r_2	72.4880
r_3	118.2919	r_3	13.2903
r_4	11.5473	r_4	87.6615
r_5	71.9173	r_5	13.8030
r_6	110.1596	r_6	84.1023
r_7	119.7562	r_7	13.8818
r_8	86.0252	r_8	78.8286
r_9	10.8213	r_9	82.0951

$F_{1,fwaf}$	Degree (°)	$F_{1,fiml}$	Degree (°)
r_1	93.5687	r_1	106.2326
r_2	82.1781	r_2	46.0083
r_3	96.1623	r_3	92.5118
r_4	83.2090	r_4	45.0289
r_5	88.9098	r_5	81.9025
r_6	8.5798	r_6	99.3001
r_7	89.9640	r_7	99.0921
r_8	8.3030	r_8	84.5021
r_9	84.5586	r_9	45.2873

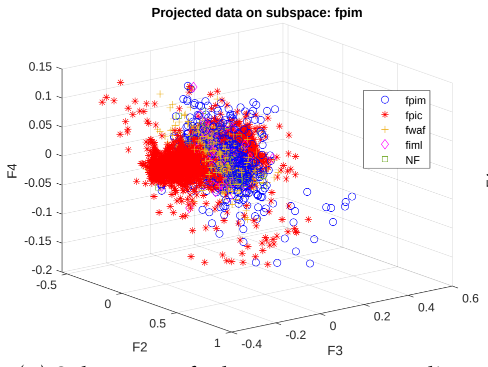
From Table 4.4 it can be stated that the ability to decouple a residual r_k is different for each fault F_i. For $F_{1,fiml}$ it seems that there are not any small angles, unlike the other faults, which indicates that no residual points closely in the direction of the fault vector. Furthermore, comparing the angle between different fault vectors gives a measure on similarity inbetween different fault classes, see Table 4.5.

***Table 4.5:** Angles between the different fault vectors.*

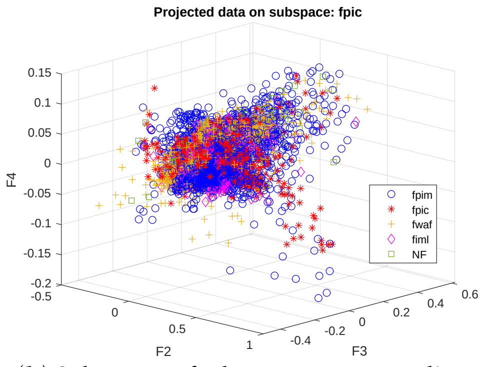
	$F_{1,fpim}$	$F_{1,fpic}$	$F_{1,fwaf}$	$F_{1,fiml}$
$F_{1,fpim}$	-	91.7837°	92.1831°	8.8465°
$F_{1,fpic}$		-	87.2201°	86.9872°
$F_{1,fwaf}$			-	87.7706°
$F_{1,fiml}$				-

According to Table 4.5, the most similar faults are f_{pim} and f_{iml} since their angle is rather small, while the others are almost orthogonal to each other.

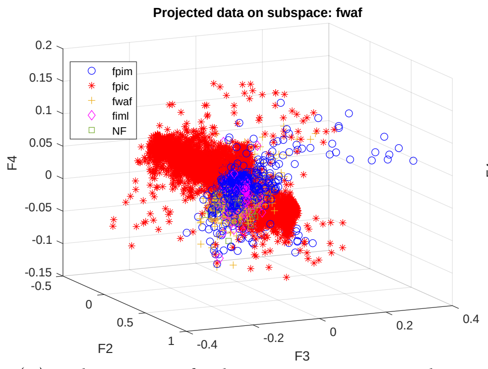
Moving on with the procedure, the training and validation data is projected onto each subspace with the transformation matrix. Figure 4.3 shows the projected training data on respective subspace where the three first direction vectors (F_2, F_3 and F_4) makes up a subset of the entire subspace.



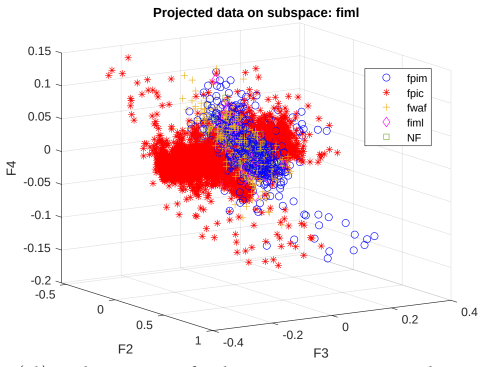
(a) *Subspace to fault vector corresponding to f_{pim}.*



(b) *Subspace to fault vector corresponding to f_{pic}.*



(c) *Subspace to fault vector corresponding to f_{waf}.*



(d) *Subspace to fault vector corresponding to f_{iml}.*

Figure 4.3: *Illustration of projected data onto subspaces. Notice that the decoupled fault class in each subspace is concentrated at the origin, which is desirable.*

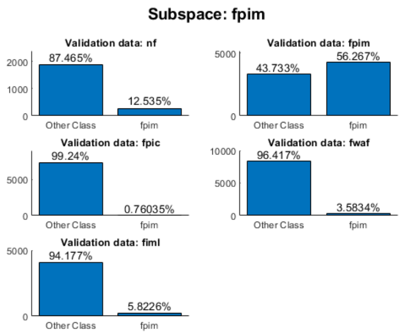
4.3 Training Classifiers

Next up, training of classifiers is made, followed by validation. Section 4.3.1 and 4.3.2 below presents binary and multi-classifiers respectively.

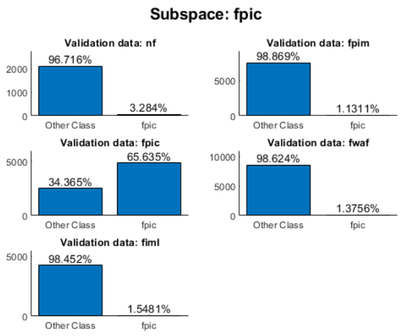
4.3.1 Binary Classifiers

The binary classifiers are trained according to an OVA strategy, where the target class is the decoupled fault and the other class consist of data from the rest faults.

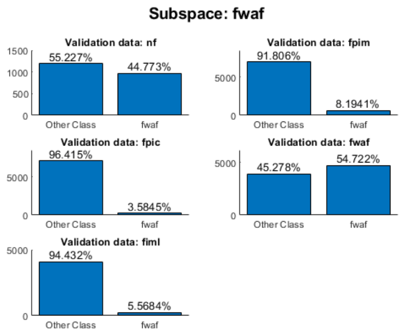
The algorithm used is SVM with RBF as kernel function. Other settings used in the algorithm are; standardizing the data, kernel scale parameter such that the software finds an appropriate value automatically and lastly using a 5% training outlier rate. The validation of these binary classifiers can be summarized in the following plots in Figure 4.4 below.



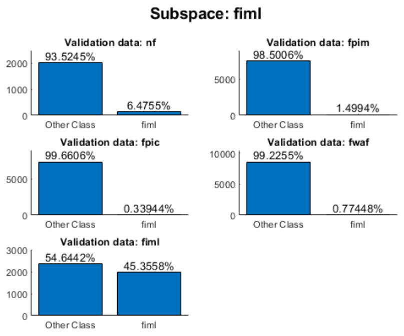
(a) *Subspace to fault vector corresponding to f_{pim}.*



(b) *Subspace to fault vector corresponding to f_{pic}.*



(c) *Subspace to fault vector corresponding to f_{waf}.*



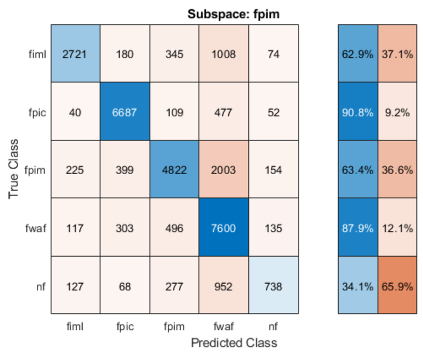
(d) *Subspace to fault vector corresponding to f_{iml}.*

Figure 4.4: *Validation of binary classifiers.*

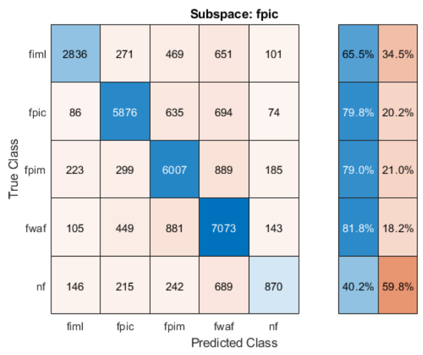
The overall performance of these classifiers from validation data seems rather promising. The main goal with these classifiers which is meant to be strived for is that each classifier should be able to identify when data comes from the decoupled fault class and when it comes from another. All plots above show good performance of distinguishing the fault classes. However, looking at the cases when validation data from the decoupled fault is tested, there is no significant confidence in the classification for the target class. One thing to notice is that f_{iml}, see Figure 4.4d, even fails to recognize the majority of validation data from its own class. This can be due to many reasons, overlapping data and overfitting are two possible reasons for this, but an important aspect however is that neither classifier reacts to data from another fault class.

4.3.2 Multi-Class Classifiers

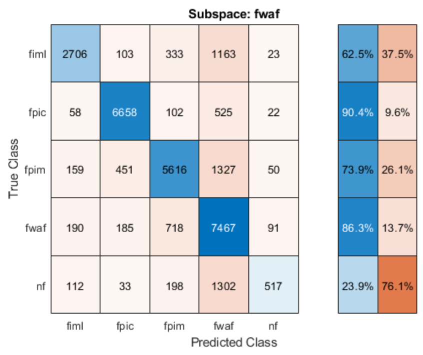
The MCC's are trained according to an OVO strategy using ECOC with SVM as learners which uses the same settings as for the binary classifiers. The validation for these classifiers are summarized in confusion matrices in Figure 4.5 below.



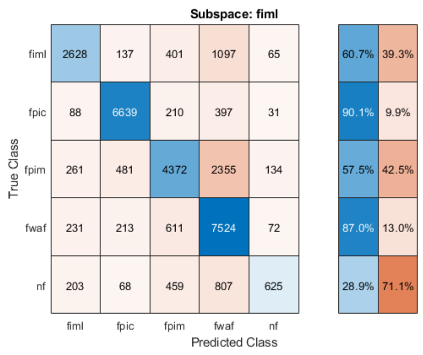
(a) *Subspace to fault vector corresponding to f_{pim}.*



(b) *Subspace to fault vector corresponding to f_{pic}.*



(c) *Subspace to fault vector corresponding to f_{waf}.*



(d) *Subspace to fault vector corresponding to f_{iml}.*

Figure 4.5: *Validation of MCC's shown in confusion matrices. The matrix to the left of a subplot shows the amount of predicted observations to each fault class in comparison to the actual true fault class. The columns to the right displays the percentage of correctly and incorrectly classified observations for each true fault class.*

The performance of these MCC's shows that each one is capable of identifying data from respective fault class. Although, they are not sufficiently strong to isolate the fault classes from each other completely. Amongst all classes, one that stands out is validation data coming from NF. As seen in Figure 4.5, all classifiers struggles in identifying validation data from NF, where most of the classifiers instead suggests f_{waf} as the most likely prediction. One conclusion to be drawn from this is that f_{waf} has some overlapping data with NF, this also can be seen in Figure 4.4c above.

An interesting aspect from these validations is that all classifiers are able to separate the decoupled fault class, i.e. the fault in question where its corresponding fault vector have been used to project data to its subspace, from NF which unlike in Section 3.1 is considered to be difficult to do. One important conclusion that can be stated from this is that the generated residual data from such complex system is not perfect. Some possible reasons for this is due to non-linearity in the system and deviations in model predictions.

Another thing worth mentioning is the interpretation of a scenario where new data is to be classified and all classifiers agrees that it belongs to respective decoupled fault class. In such case, the new data would be considered belonging to NF. This would be an alternative solution to identify fault-free data when the classifiers struggles to do so solely.

4.4 Data Classification

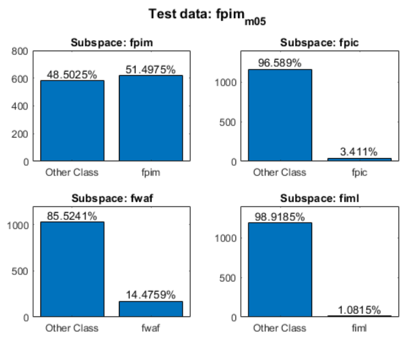
The data classification is performed as mentioned in Section 3.6.2. Using the described classifier selection (see Algorithm 1) when aggregating decision outputs from multiple classifiers, the number nearest neighbors K searched is set to 12 for all cases. The considered test data used to evaluate the proposed framework are:

- $\{f_{pim,-5\%}, f_{pim,+5\%}, f_{pic,-5\%}, f_{waf,-5\%}, f_{iml,4mm}\}$ as known fault classes.
- $\{f_{acl,4mm}, f_{bcl,4mm}, f_{af,th15}\}$ as unknown fault classes.

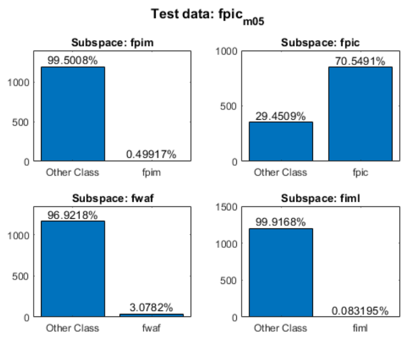
The motivation for these decisions of test data is because they fall under the category *small fault sizes* and are therefore harder to classify than larger fault sizes. Sections 4.4.1 and 4.4.2 below present the diagnostic framework using binary and MCC's respective.

4.4.1 Binary Classifiers

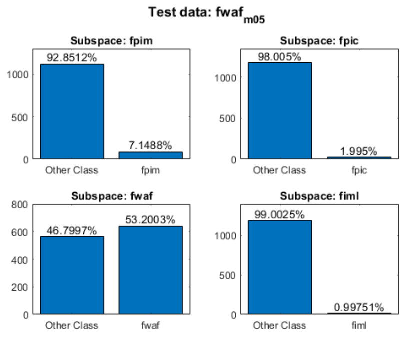
The classification performance of the binary classifiers is first evaluated separately and then by fusing multiple classifiers.



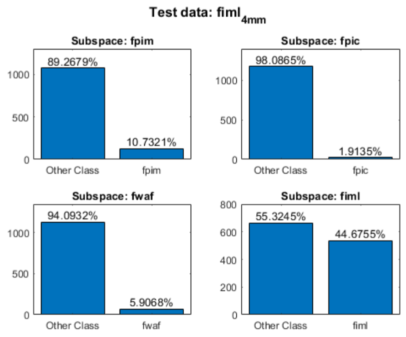
(a) *Test data $f_{pim,-5\%}$.*



(b) *Test data $f_{pic,-5\%}$.*



(c) *Test data $f_{waf,-5\%}$.*



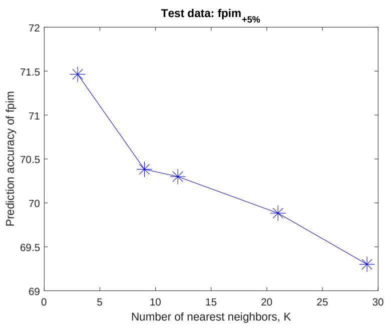
(d) *Test data $f_{iml,4mm}$.*

Figure 4.6: *Test data classified separately with binary classifiers.*

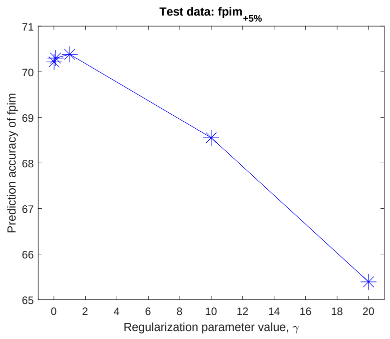
In Figure 4.6 it can be seen that the binary classifiers excellently rejects data not belonging to the target class. However, it can also be stated that they struggle to clearly distinguish when data actually belongs to the target fault, except for f_{pic}'s binary classifier in Figure 4.6b. Besides, Figure 4.6d even shows that the binary classifier for f_{iml} is not capable to single handedly classify the majority of sampels, thus one conclusion to be drawn from this is that there is a need to take more than one classifier in consideration in the decision making.

Fusing binary classifiers

Performing the multiple classifier weighting described in Section 3.4.2 requires to choose the parameters K and γ. This was done by fixing one while varying the other. Figure 4.7 below shows how the classifier accuracy for predicting f_{pim} varies for different parameter values, when test data $f_{pim,+5\%}$ is used.



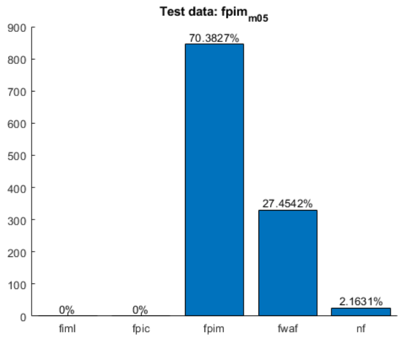
(a)



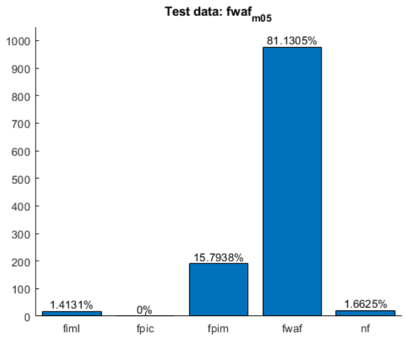
(b)

Figure 4.7: *Classifier accuracy plotted against different K and γ values. In 4.7a $\gamma = 1$ and in 4.7b $K = 9$.*

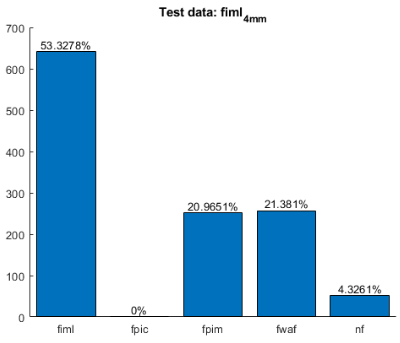
From Figure 4.7 it can be seen that there seems to exist some optimal values in terms of maximizing the classification accuracy. An excessive value of one parameter tends to decrease the performance. Therefore these parameters play an important roll and the values must be well-balanced. Although these values perhaps should be adjusted for each test data, as a simplification they were chosen as $K = 9$ and $\gamma = 1$, which is based on the figure above but also from experience with other test data. Figure 4.8 shows the results of using these parameters and weighting the decisions. It truly shows the strength and advantage combined multiple classifiers have over a single classifier.



(a) *Test data $f_{pim,-5\%}$.*



(b) *Test data $f_{waf,-5\%}$.*

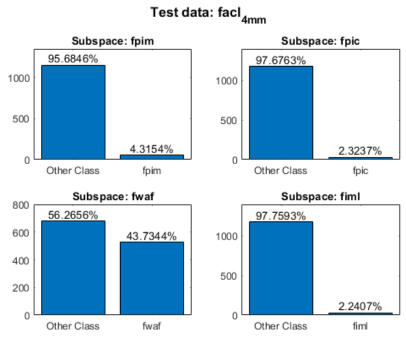


(c) *Test data $f_{iml,4mm}$.*

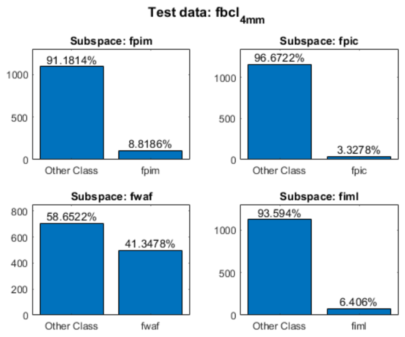
Figure 4.8: *Plots showing results when using the proposed diagnostic framework for the binary classifiers. Test data for $f_{pic,-5\%}$ is not showed since it was singely passed through the selection phase.*

Unknown fault classes

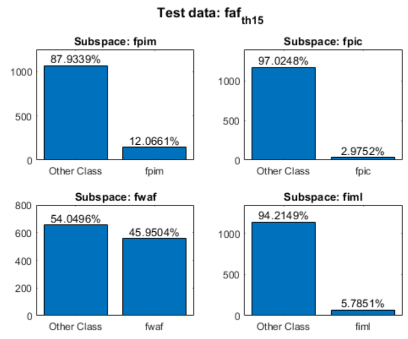
Lastly, classification of the unknown fault classes is presented in Figure 4.9 below.



(a) Test data $f_{acl,4mm}$.



(b) Test data $f_{bcl,4mm}$.



(c) Test data $f_{af,th15}$.

Figure 4.9: *Plots of classification performance for the unknown fault classes.*

In Figure 4.9 it can be stated that the binary classifiers agrees the three unknown fault classes does not belong to f_{pim}, f_{pic} or f_{iml}. Although, they recognizes about 40% – 45% of the samples to f_{waf}. Looking at the angles between the fault vector from the known faults and respective unknown fault class, see Table 4.6, it can be seen that $F_{1,fwaf}$ differs greatly from the rest since the fault vectors are not far from being parallel. Hence, this indicates that they are hard to isolate from f_{waf} and is the reason why its binary classifier recognized samples from these unknown faults.

Table 4.6: Angle between the fault vectors. The gray marked row is meant to highlight that $F_{1,fwaf}$ differs from the rest.

	$F_{1,facl}$	$F_{1,fbcl}$	$F_{1,faf}$
$F_{1,fpim}$	89.4845°	93.5328°	84.6897°
$F_{1,fpic}$	83.2676°	85.5862°	86.8894°
$F_{1,fwaf}$	5.5024°	3.3753°	7.8651°
$F_{1,fiml}$	84.8511°	89.1239°	80.2939°

4.4.2 Multi-Class Classifiers

Likewise the binary classifiers, the classification performance is first evaluated separately and then by fusing multiple classifiers. Tables 4.7-4.10 below presents the classification done separately.

Table 4.7: Classification performance for each MCC when classifying test data $f_{pim,-5\%}$.

Fault class / Subspace	NF	f_{pim}	f_{pic}	f_{waf}	f_{iml}
f_{pim}	2.75%	58.82%	4.41%	32.45%	1.58%
f_{pic}	3.58%	64.14%	8.49%	23.045%	0.75%
f_{waf}	0.92%	61.56%	3.49%	33.69%	0.33%
f_{iml}	1.91%	59.32%	2.91%	33.19%	2.66%

Table 4.8: Classification performance for each MCC when classifying test data $f_{pic,-5\%}$.

Fault class / Subspace	NF	f_{pim}	f_{pic}	f_{waf}	f_{iml}
f_{pim}	0.25%	2.25%	88.85%	7.57%	1.08%
f_{pic}	0.33%	5.82%	82.61%	8.40%	2.83%
f_{waf}	0.25%	1.33%	87.69%	8.32%	2.41%
f_{iml}	0.17%	3.16%	87.60%	8.15%	0.915%

Table 4.9: *Classification performance for each MCC when classifying test data $f_{waf,-5\%}$.*

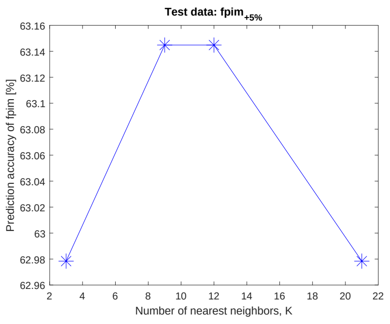
Subspace \ Fault class	NF	f_{pim}	f_{pic}	f_{waf}	f_{iml}
f_{pim}	1.83%	9.23%	4.32%	81.38%	3.24%
f_{pic}	2.66%	9.98%	10.72%	72.15%	4.49%
f_{waf}	1.50%	9.23%	2.49%	84.04%	2.74%
f_{iml}	0.75%	12.47%	3.91%	79.05%	3.83%

Table 4.10: *Classification performance for each MCC when classifying test data $f_{iml,4mm}$.*

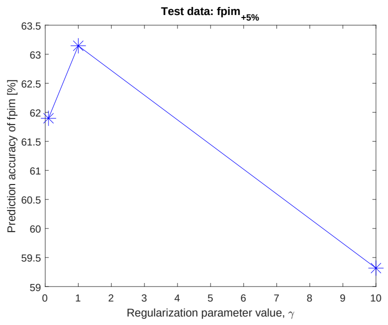
Subspace \ Fault class	NF	f_{pim}	f_{pic}	f_{waf}	f_{iml}
f_{pim}	0.25%	12.73%	3.08%	24.38%	59.57%
f_{pic}	1.08%	12.90%	6.74%	15.89%	63.39%
f_{waf}	0.50%	11.31%	2.08%	24.04%	62.06%
f_{iml}	0.92%	9.57%	2.25%	27.62%	59.65%

Fusing multi-class classifiers

Again, the parameters in the optimization problem are found similarly as for the binary classifiers.



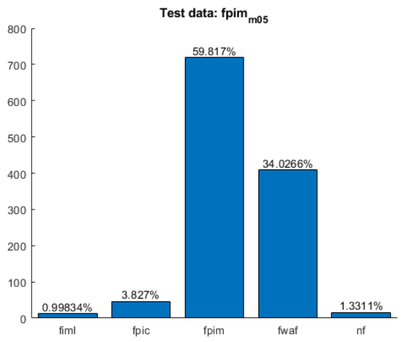
(a)



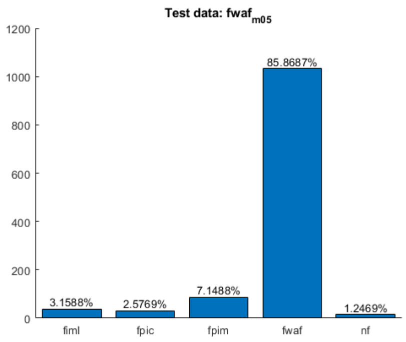
(b)

Figure 4.10: *Classifier accuracy plotted against different K and γ values. In 4.10a $\gamma = 1$ and in 4.10b $K = 9$.*

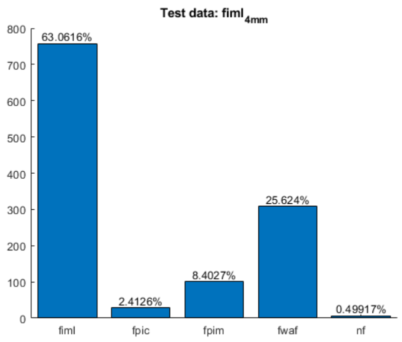
Figure 4.10 it can be seen that even for the MCC the parameters follow a similar pattern as for the binary classifiers and that they should be chosen carefully. By looking at the plots and with experience from testing other test data, the parameters are set to $K = 9$ and $\gamma = 1$ when fusing multiple classifiers, see Figure 4.11 below.



(a) Test data $f_{pim,-5\%}$.



(b) Test data $f_{waf,-5\%}$.



(c) Test data $f_{iml,4mm}$.

Figure 4.11: *Plots showing results when using the proposed diagnostic framework for the MCC's. Test data for $f_{pic,-5\%}$ is not showed since it was singely passed through the selection phase.*

The results from Figure 4.11 shows promising performance. Comparing with the classifications in Tables 4.7-4.10 the accuracy does not seem to increase as much as the binary case, but there is a clear agreement amongst them.

One important part of the proposed diagnostic framework which is yet to be evaluated is the possibility to identify if new test data comes from the fault-free case. Since the boundary between data from a small fault size and data from fault-free scenario could be hard to distinguish, it is extremely important to identify any of these in order to improve the life quality of the engine and prevent a fault from increasing over time. One suggested way to test this is to classify with the MCC's. Either all classifiers agrees that NF is the most likely case or all classifiers suggest that the data belongs to their decoupled fault class. Table 4.11 below shows the result when test data from NF is used.

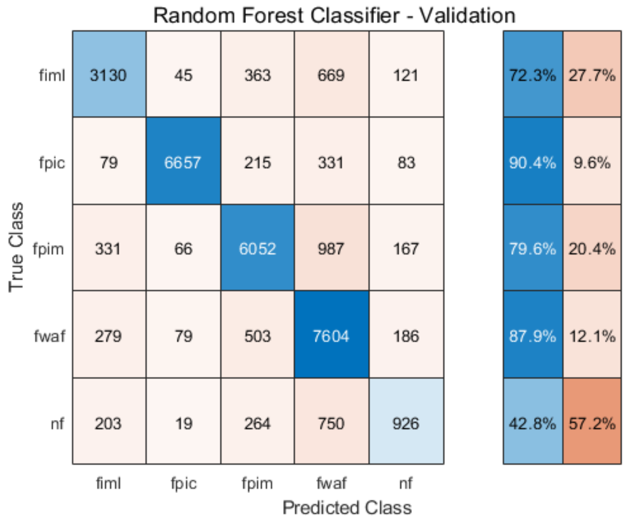
Table 4.11: *Classification performance for each MCC when classifying test data NF.*

Subspace \ Fault class	NF	f_{pim}	f_{pic}	f_{waf}	f_{iml}
f_{pim}	67.94%	5.66%	1.42%	22.40%	2.58%
f_{pic}	72.19%	4.50%	3.75%	17.32%	2.25%
f_{waf}	59.78%	3.91%	0.83%	33.22%	2.25%
f_{iml}	63.78%	14.82%	2.50%	14.65%	4.25%

Table 4.11 shows that the MCC's are fully capable of classifying data belonging to the NF case where there is a clear agreement.

4.4.3 Comparing with Random Forest

Training was done with random forest classifier on data from all fault classes, which yielded a MCC. Settings used were default for classification with *TreeBagger* function in Matlab (available in the *Statistics and Machine Learning toolbox*) with 100 bagged trees. Validation can be seen in Figure 4.12. As noticed, the overall validation performance is good for all fault classes and the accuracy on NF is most of the times higher than the ones in Figure 4.5. Evaluation of the test data can be found in Table 4.12. It can be seen that the random forest classifier is able to make confident predictions for challenging test data.



True Class	fiml	fpic	fpim	fwaf	nf		
fiml	3130	45	363	669	121	72.3%	27.7%
fpic	79	6657	215	331	83	90.4%	9.6%
fpim	331	66	6052	987	167	79.6%	20.4%
fwaf	279	79	503	7604	186	87.9%	12.1%
nf	203	19	264	750	926	42.8%	57.2%

Figure 4.12: *Validation of Random Forest classifier.*

Table 4.12: *Classification performance of test data on random forest classifier.*

Test data \ Fault class	NF	f_{pim}	f_{pic}	f_{waf}	f_{iml}
$f_{pim,-5\%}$	3.41%	64.73%	1.75%	28.54%	1.58%
$f_{pic,-5\%}$	0.92%	3.16%	88.44%	3.91%	3.58%
$f_{waf,-5\%}$	3.16%	9.89%	1.75%	76.56%	8.65%
$f_{iml,4mm}$	1.91%	8.65%	0.75%	20.80%	67.89%

To summarize the performances of the different classifiers, the accuracies are presented in Table 4.13 below.

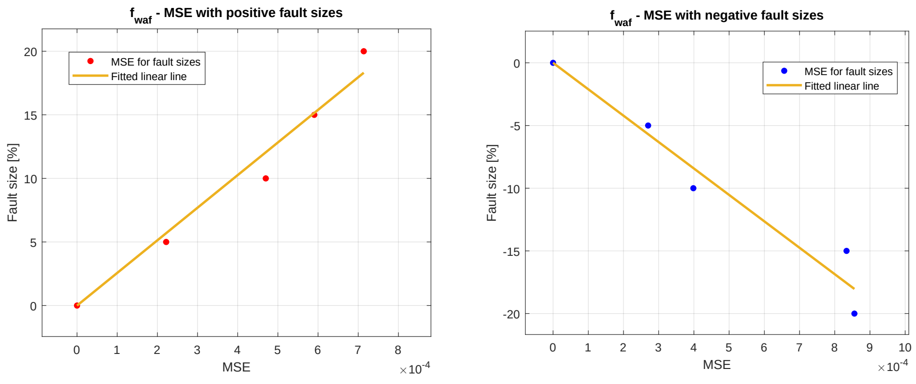
Table 4.13: *Summary of accuracies for the different classifiers with the mentioned test data (Binary and MCC refers to the proposed classifiers when an information fusion is performed). The highlighted cells indicate the type of classifier yielding highest accuracy for predicting the test data in question.*

Test data \ Classifier	Binary	MCC	R.F
$f_{pim,-5\%}$	70.38%	59.82%	64.73%
$f_{waf,-5\%}$	81.13%	85.87%	76.56%
$f_{iml,4mm}$	53.33%	63.06%	67.89%

A conclusion to be drawn from this is that a random forest classifier is more robust than any of the singly classifiers mentioned previously. However, using the proposed framework and combining the classifiers results in performances which relates well to random forest classifier. Some fault classes receives higher accuracy while others tend to be more difficult to classify when comparing to the state-of-art. Using the proposed framework have of course the possibility to identify if data comes from an unknown fault while a random forest classifier is limited to always produce a prediction of any known fault class. Although, making comparisons from a time perspective there is no doubt that random forest classifiers are preferable (at least in this study). Therefore, it would be considered a strong classifier to begin any analysis with, which delivers fast execution time and high classification performance.

4.4.4 Fault Size Estimation

Modeling a linear function that estimates fault size based on MSE for a given residual data set naturally yields better fitted line when data from several fault sizes is available. To set a reference, consider Figure 4.13 where all fault sizes for f_{waf} have been included. Worth mentioning is that this has been done for the remaining fault classes and the results show similar performance.



(a) *(b)*

Figure 4.13: *Fault size plotted against MSE for fault f_{waf}, where Figure 4.13a shows for positive fault sizes and Figure 4.13b for negative fault sizes. The yellow line is the linear model fitted to the generated data points.*

As shown in Figure 4.13, both directions of the fitted line seems to follow a linear pattern. One thing to keep in mind using this type of estimation with MSE is that the estimation of fault sizes, at least for the sensor faults, should be consistent on both sides of which sign the fault size belongs to, i.e. it would be unexpected that an estimation on new data suggests $+5\%$ and -20% severity. A suggested approach would be to look at both sides, positive and negative fault sizes, and verify that an estimation lies within reasonable limits.

Evaluating the method on leakage fault f_{iml}, where two data set comes from unknown fault sizes yields following result in Figure 4.14 below.

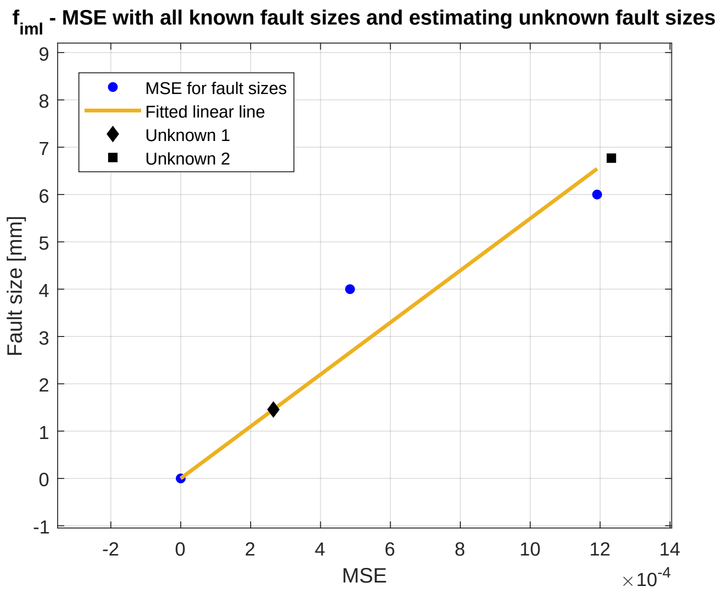


Figure 4.14: *Fault size plotted against MSE for f_{iml} where the two unknown data sets are estimated.*

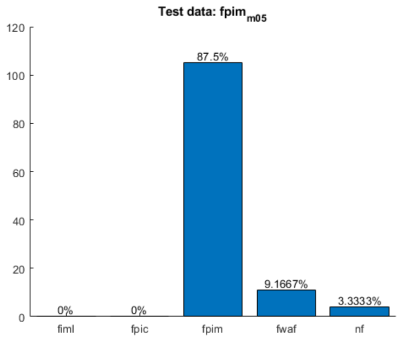
According to this fitted linear line, *Unknown 1* and *Unknown 2* have an approximate hole diameter of 1.5 mm and 6.7 mm respective. Unlike sensor faults, f_{iml} has only positive fault sizes, which leads to relying on a single estimation. Verifying whether these estimations are reasonable can be done comparing corresponding data to the existing ones with known fault sizes.

4.5 Preprocessing Data by Reshape

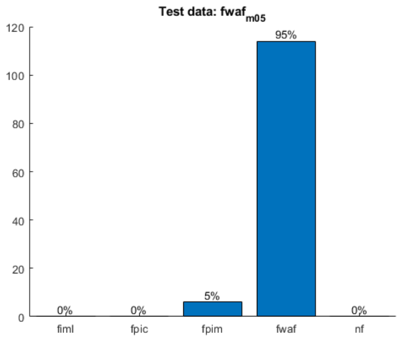
In this section, evaluation is made likewise previously sections but this time while preprocessing the data as described in Section 3.2 in Chapter 3. Reshaping is done with data from each fault size separately along with partitioning into training and validation, to finally assemble all training and validation data corresponding to fault i. The same parameters K and γ are used. The number of batches to divide each residual in were set to 10, which reshapes the residual data to 90 columns.

4.5.1 Binary Classifiers - Prediction

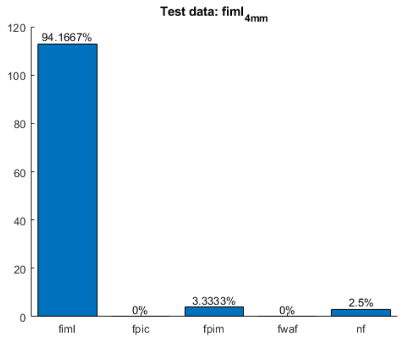
Figure 4.15 below shows the results using the proposed framework with the binary classifiers.



(a) Test data $f_{pim,-5\%}$.



(b) Test data $f_{waf,-5\%}$.



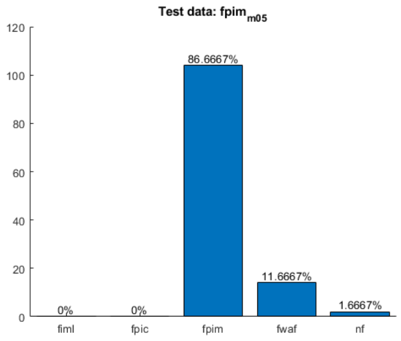
(c) Test data $f_{iml,4mm}$.

Figure 4.15: *Plots showing results when using the proposed diagnostic framework for the binary classifiers. Test data for $f_{pic,-5\%}$ is not showed since it was singely passed through the selection phase.*

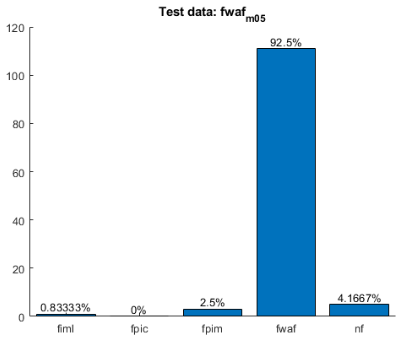
Evaluating test data on these classifiers yields significantly higher accuracy and there is overall a higher confidence in the decision making. Already in the validation major differences in performance can be seen, where there are clearer separations in prediction of target class and outliers, in comparison to previously validation plots. However, this have shown to come with a downside, where the setback lies in classifying data from unknown fault classes. Preprocessing the residual data by reshaping as in this manner has shows making it more difficult to identify unknown data from fault classes with the binary classifiers.

4.5.2 Multi-Class Classifiers - Prediction

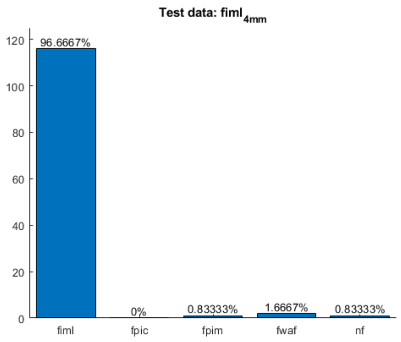
Figure 4.16 below shows the results using the proposed framework with the MCC's.



(a) Test data $f_{pim,-5\%}$.



(b) Test data $f_{waf,-5\%}$.



(c) Test data $f_{iml,4mm}$.

Figure 4.16: *Plots showing results when using the proposed diagnostic framework for the MCC's. Test data for $f_{pic,-5\%}$ is not showed since it was singely passed through the selection phase.*

Likewise the binary classifiers, the classification performance for the MCC's increases for test data known to belong to any of the existing fault classes in the training data.

5 Discussion

This chapter discusses the proposed method and the experimental results presented in Chapter 4.

5.1 Results

The results in Chapter 4 show promising performance where there is a clear difference when comparing the decisions made by a single classifier and multiple classifiers fused together. One thing to bear in mind is that the tested fault scenarios have been with the smallest fault sizes available, making it the most difficult yet interesting case. Larger fault sizes has been tested and shown to be classified correctly with higher accuracy.

The validation for the binary classifiers in Figure 4.4, shows that in each subspace there is an expert classifier which is capable of identifying its target class, the decoupled fault class, and reject data from other fault classes. However, based on the proportion of samples that are correctly classified as the target class it is not always a clear separation. It would be desirable to achieve a higher confidence in the classification of the target class in each subspace, but working with such data where a fault class has different realizations due to different fault sizes it comes with a price. It has been evaluated with OCC's (as in Section 3.1) that data from the fault classes has a strong overlapping, i.e. that an OCC for a certain fault can explain and capture data from another fault with a high classification rate. An OCC aims at capturing the unique properties of the target class with no counterexamples, unlike binary classifiers which aims at minimizing the error between the other class, making OCC far more flexible. Although, in this type of problem there is a need to distinguish the classes apart. Hence, there are two aspects of constructing this kind of set of binary classifiers. Either one makes the classifiers good at just classifying the target class or one focuses on making the classifiers good at rejecting outliers to the target class. This would refer to the effects of overfitting. In this study, the latter has shown that an aggregation yields a higher accuracy in the decision making than a single classifier.

The validation for the MCC's in Figure 4.5 shows that some fault classes are easier to classify than other, e.g. f_{pic} can clearly be distinguished from other classes even at a small fault size while f_{iml} struggles do so. In all subspaces, f_{pic} is easiest to distinguish and the for the others it varies. One interesting thing to acknowledge is the relationship between f_{pim} and f_{iml} throughout the validation but also the test data. The classifiers are able to separate data from these classes even though the angular difference between their fault vectors is small. Looking at the performance of the MCC's for the test data, it can be seen that each classifier is capable at predicting the test data as the most likely fault class. If a simple majority vote among them based on assigning the fault class with highest score, it would result in a clear agreement.

Comparing the results from random forest classifier with the ones received from the proposed diagnostic framework show how well the performance is compared to a reference. The MCC's relates well to the reference (see Table 4.12) where they are slightly better or worse in predicting accurately depending on test data. The downside of course with combining MCC's as done in this study over a more simpler classifier as random forest is the time complexity. Not only increases the time it takes to train the SVM models significantly, but also the KNN search leads to increased time consumption (both in the classifier selection stage and in the fusing stage). The binary classifiers faces similar time complexity problem.

Reshaping the residual data has shown to be an interesting preprocess technique. Its advantages can be summarized in increased accuracy of classifying test data from known fault classes and reduced time consumption from training the classifiers to weighting the decision outputs. However, it comes with a setback where the binary classifiers struggles to identify that data comes from an unknown fault class. Using this technique in the proposed diagnostic framework would only be appropriate when the test data is confident to belong to any of the known fault classes in the training data. Another thing to point out regarding the use of this is whether it is consistent for all drive cycles or needs to be adjusted according to every new measurement. Since the engine operates at different operating points throughout the measurements, it could lead to imbalance between the different batches. This has not been studied in this thesis and would be required to tell if this technique could be applied in general cases.

The proposed fault size estimation using MSE as a measure of how much residual data deviates from the origin has shown to be simple yet effective solution. The linear approximation between different fault sizes gets more precise when taking several data sets into account, i.e. more realizations of a fault improves the estimation on new data. The fault size estimations on the two unknown fault sizes from f_{iml}, see Figure 4.14, are reasonable since it is known that data from

Unknown 1 have been injected with smaller severity than *Unknown 2* during the measurements of the data. Another verification is that the excitements of residuals in *Unknown 2* resembles the ones in $f_{iml,6mm}$, while *Unknown 1* has the lowest of all available f_{iml} data.

5.2 Methodology

Decoupling a fault class using PCA has shown to be a simple yet effective method. It allows to measure the similarity between data sets with their fault vector (and angles between subspaces), which does not come with further complexity. Although, working with PCA there are some crucial properties in the data that affects this method, such as how strong correlation there is between features and how well-distributed variance there is. For instance, if a data set has large variance it would lead to reduced total explained variance in the first principle component. Hence, projecting the data onto the subspace corresponding to its fault vector would get more spread out and the idea regarding decoupling a fault would not work as preferred.

The classifier selection algorithm worked smoothly and were able to locate the most competent (known) classes to the test data for any fault size in the training data. The threshold parameter J and K were shown to be suitable values for the task. A larger K-value would only be beneficial for test data belonging to small fault sizes since they are most difficult to distinguish (especially for overlapping data) and the cost of it would increase the time consumption. One interesting aspect with this selection method that could be of interest for further study is to instead of searching for a fix number of neighbors, to search for nearest neighbors with a radius from each test sample. It would for instance open up the possibility to identify unknown fault classes in cases where the search does not find any neighbors.

Taking multiple classifier into consideration in the decision making truly has the potential to improve the classifications. Solving the optimization problems for binary and MCC's as presented in Section 3.4 works on certain conditions in order to achieve high quality decisions. One of which is that the local competent classifiers should be accurate and diverse. Having a set of classifiers that make independent errors is desirable since it would improve the classification performance, as seen with the binary classifiers. Furthermore, an important thing to keep in mind with the used aggregation method is that the objective function highly relies on the class index vector $\mathbf{t}(\boldsymbol{x}_k)$. Since the quadratic term has greater impact on finding the optimal weights it is therefore beneficial to reduce overlapping data in order for $\mathbf{t}(\boldsymbol{x}_k)$ to find correct true class more frequently.

Regarding the aggregation method from a time perspective, some of the things affecting it is the chosen K-value and the number of classifiers which decisions outputs are to be weighted. For the binary classifiers, changing K solely from 3 to 21 increases the consumed time with approximately 150 seconds, while for the MCC's it increases by 950 seconds. An increased number of local classifiers on the other hand enlarges the **A** matrix but also the dimension on $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and $\boldsymbol{\eta}$, leading to increased time when solving the objective function for each sample in the test data. Nevertheless, aiming for a fast functional system requires to find a sufficient low K-value and only the necessary competent classifiers.

6 Conclusion

This chapter contains a summarization of the purpose and research questions along with a brief section regarding future study.

In this work, a novel method inspired from model-based fault diagnostics has been integrated in a MCS which is part of a systematical diagnostic framework. By decoupling data using PCA generated from residuals that detects any inconsistencies with the fault-free case, decision outputs from trained classifiers are weighted to form a final diagnosis statement. For each decoupled fault there exist a subspace where data from the other fault classes are projected onto. The idea itself of decoupling faults in separate subspaces has proven to be one way to achieve diversity between different classifiers, something which is sought for in a MCS. Two types of classifiers have been used to recognize patterns from the different fault classes in these subspaces: binary and MCC's. The former's primary objective is to identify data from unknown fault classes but has shown to be a promising alternative for decision making when fusing decisions from multiple binary classifiers. The latter aims at classifying known fault classes and importantly to also identifying if data belongs to the fault-free case.

To tell how well the classification performance is for both mentioned types, a comparison has been made with a random forest classifier. Results indicate that the proposed diagnostic framework relates well compared to state-of-the-art, where challenging test data has been evaluated. How easy it is to distinguish two classes apart can be concluded to depend on how the angle between fault vectors relates, where a small angle suggest it is more difficult. The smallest angle amongst the known faults in the training data was found to be between f_{pim} and f_{iml}, but separating data from respective fault showed to work without any problem. Detecting unknown fault classes, i.e. f_{acl}, f_{bcl} and f_{af}, on the other hand were especially difficult to distinguish from f_{waf}, since the angles their fault vector and f_{waf}'s are small. One conclusion that can be drawn when it comes to detecting unknown fault classes is that comparing the angle between different fault vectors could tell how easy it is, but does not necessarily mean that a small angle suggest that the data originates from the same sensor. This could instead be an indirect cause of correlated residuals. Another analysis that can be drawn from the fault vectors is the angle between base-residuals. It suggests how relevant a residual is to the fault in question, which points out which residuals that could be removed while minimizing the loss of information.

Taking multiple observations into account when training the classifiers can be done in multiple ways. The presented method has shown to increase the classification performance of known fault classes while identifying unknown faults gets more difficult. However, it is not claimed to work for measured data from different drive cycles or in other applications.

6.1 Future Work

This work has shown to be an interesting topic when working with fault diagnostics for an internal combustion engine. Not all aspects has been covered or analysed and therefore some relevant future work can be studied further.

Firstly, a feature selection of the most informative residuals for each fault class would be interesting to use in the proposed diagnostic framework. This could for instance be done using random forest or alternatively with a correlation analysis. This would show if a dimension reduction is possible and still maintain robust classification performance.

Secondly, to further develop the reshaping method. An interesting approach would be to make it independent on the current drive cycle. Rather than generating batches by dividing residuals after different time sequences, to instead generate batches from random observations and try to capture different operating points in such manner.

Thirdly, to evaluate this framework on other drive cycles, another set of available training data and other applications.

Bibliography

[1] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[2] Vladimir Bondarenko. drawla - draw toolbox for linear algebra. `https://www.mathworks.com/matlabcentral/fileexchange/23608-drawla-draw-toolbox-for-linear-algebra`, May 2021.

[3] Alceu S Britto Jr, Robert Sabourin, and Luiz ES Oliveira. Dynamic selection of classifiers—a comprehensive review. *Pattern recognition*, 47(11):3665–3680, 2014.

[4] Steven Brunton and J. Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 01 2019. ISBN 9781108422093. doi: 10.1017/9781108380690.

[5] Hakan Cevikalp and Robi Polikar. Local classifier weighting by quadratic programming. *IEEE transactions on neural networks*, 19(10):1832–1838, 2008.

[6] Rafael MO Cruz, Robert Sabourin, and George DC Cavalcanti. Dynamic classifier selection: Recent advances and perspectives. *Information Fusion*, 41:195–216, 2018.

[7] Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Arti
cial Intelligence Research 2*, 2:263–286, 1995.

[8] Lars Eriksson, Simon Frei, Christopher Onder, and Lino Guzzella. Control and optimization of turbocharged spark ignited engines. *IFAC Proceedings Volumes*, 35(1):283–288, 2002.

[9] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[10] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. " O'Reilly Media, Inc.", 2017.

[11] Michael Grant and Stephen Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pages 95–110. Springer-Verlag Limited, 2008. `http://stanford.edu/~boyd/graph_dcp.html`.

[12] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. `http://cvxr.com/cvx`, March 2014.

[13] Jiawei Han, Micheline Kamber, and Jian Pei. Data mining concepts and techniques third edition. *The Morgan Kaufmann Series in Data Management Systems*, 5(4):83–124, 2011.

[14] Anshul Jindal. Dimensionality reduction using pca on multivariate timeseries data. `https://medium.com/@ansjin/dimensionality-reduction-using-pca-on-multivariate-timeseries-data-b5cc07238dc4`, 2019. Online; accessed 22-January-2021.

[15] Daniel Jung. Residual generation using physically-based grey-box recurrent neural networks for engine fault diagnosis. *arXiv preprint arXiv:2008.04644*, 2020.

[16] Bartosz Krawczyk, Michał Woźniak, and Francisco Herrera. On the usefulness of one-class classifier ensembles for decomposition of multi-class problems. *Pattern Recognition*, 48(12):3969–3982, 2015.

[17] Bartosz Krawczyk, Mikel Galar, Michał Woźniak, Humberto Bustince, and Francisco Herrera. Dynamic ensemble selection for multi-class classification with one-class classifiers. *Pattern Recognition*, 83:34–51, 2018.

[18] Ludmila I Kuncheva, James C Bezdek, and Robert PW Duin. Decision templates for multiple classifier fusion: an experimental comparison. *Pattern recognition*, 34(2):299–314, 2001.

[19] Andreas Lundgren and Daniel Jung. Data-driven open set fault classification and fault size estimation using quantitative fault diagnosis analysis. *arXiv preprint arXiv:2009.04756*, 2020.

[20] Aizhong Mi, Lei Wang, and Junyan Qi. A multiple classifier fusion algorithm using weighted decision templates. *Scientific Programming*, 2016, 2016.

[21] Gang Niu, Tian Han, Bo-Suk Yang, and Andy Chit Chiow Tan. Multi-agent decision fusion for motor fault diagnosis. *Mechanical Systems and Signal Processing*, 21(3):1285–1299, 2007.

[22] John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[23] Oriol Pujol, Petia Radeva, and Jordi Vitria. Discriminant ecoc: A heuristic method for application dependent design of error correcting output codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28 (6):1007–1012, 2006.

[24] Romesh Ranawana and Vasile Palade. Multi-classifier systems: Review and a roadmap for developers. *International Journal of Hybrid Intelligent Systems*, 3(1):35–61, 2006.

[25] Anderson Rocha and Siome Klein Goldenstein. Multiclass from binary: Expanding one-versus-all, one-versus-one and ecoc-based approaches. *IEEE Transactions on Neural Networks and Learning Systems*, 25(2):289–302, 2013.

[26] Lior Rokach. Decision forest: Twenty years of research. *Information Fusion*, 27:111–125, 2016.

[27] Terry Windeatt and Reza Ghaderi. Coding and decoding strategies for multi-class learning problems. *Information Fusion*, 4(1):11–21, 2003.

[28] Tomasz Woloszynski and Marek Kurzynski. A probabilistic model of classifier competence for dynamic ensemble selection. *Pattern Recognition*, 44 (10-11):2656–2668, 2011.

[29] Tomasz Woloszynski, Marek Kurzynski, Pawel Podsiadlo, and Gwidon W Stachowiak. A measure of competence based on random classification for dynamic ensemble selection. *Information Fusion*, 13(3):207–213, 2012.

[30] Kevin Woods, W. Philip Kegelmeyer, and Kevin Bowyer. Combination of multiple classifiers using local accuracy estimates. *IEEE transactions on pattern analysis and machine intelligence*, 19(4):405–410, 1997.