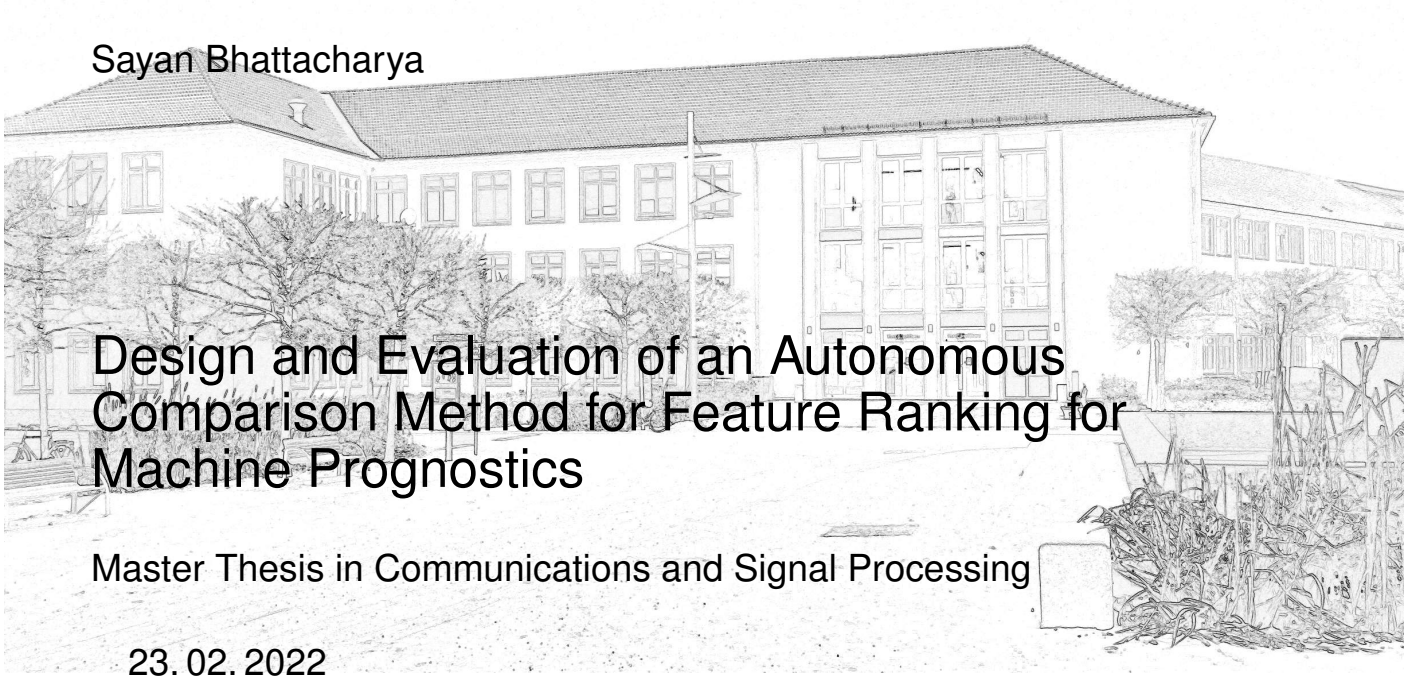
TECHNISCHE UNIVERSITÄT
ILMENAU

Communication Networks Group

Sayan Bhattacharya



Design and Evaluation of an Autonomous Comparison Method for Feature Ranking for Machine Prognostics

Master Thesis in Communications and Signal Processing

23. 02. 2022

Please cite as:
Sayan Bhattacharya, “Design and Evaluation of an Autonomous Comparison Method for Feature Ranking for Machine Prognostics,” Master Thesis (Masterarbeit), Technische Universität Ilmenau, Deptpartment of Electrical Engineering and Information Technology, March 2022.

Technische Universität Ilmenau
Department of Electrical Engineering and Information Technology
Communication Networks Group

Helmholtzplatz 2 · 98693 Ilmenau · Germany

http://www.tu-ilmenau.de/kn

Design and Evaluation of an Autonomous Comparison Method for Feature Ranking for Machine Prognostics

Master Thesis in Communications and Signal Processing

submitted by

Sayan Bhattacharya

born on 05. January 1993
in Kalkutta, Indien

in the

Communication Networks Group

Deptartment of Electrical Engineering
and Information Technology
Technische Universität Ilmenau

Advisors : **Prof. Dr. rer. Nat. Jochen Seitz,**
Prof. Dr. -Ing. Patrick Maeder,
Msc. Umut Onus,
Msc. Dominik Walther

Submission Date : 23. 02. 2022

Declaration

I declare that the work is entirely my own and was produced with no assistance from third parties.
I certify that the work has not been submitted in the same or any similar form for assessment to any other examining body and all references, direct and indirect, are indicated as such and have been cited accordingly.

(Sayan Bhattacharya)
Ilmenau, 22. März 2022

Acknowledgements

A brief section of this thesis cannot do justice to how important people and ideas can be in one's journey. It is, however, what I can present, and offer it I shall.

I would like to thank my family for their unconditional support. My parents encouraged me eagerly to follow through with my desire of studying abroad and offer their support in overcoming the time zone difference.

I am grateful to my friends scattered all around this pale blue dot for lending an ear and asking for one whenever needed. It means a lot. As for my friends in Ilmenau, thank you for making this place warm and genuinely friendly.

After working for around seven months at the IMMS System design group, an expression of gratitude to its members is overly due. Special thanks to Prof. Seitz and Prof. Maeder and their group for voicing their enthusiasm in my collaboration with the group and prompting me to think about my future beyond academics. I would like to thank Umut for humoring me with an inordinate amount of his time and feedback. I want to thank Dr. Krug for helping me to think clearly. Working with Umut and everyone else in the group has been an excellent growth experience and an opportunity that I'm fortunate to have been presented with.

Lastly, I want to thank Keith Jarret and his work, *A Multitude of Angels*, to keep me lively with his organic solo piano improvisations during the dark long winters of Ilmenau.

Abstract

The field of Feature Ranking (FR) deals with the task of finding a hierarchy of useful features in a given dataset. One useful practical application of FR can be in the field of predictive maintenance (PdM), where the health of the machine is constantly monitored. As data-driven approaches become more popular, bringing Artificial Intelligence (AI) solutions to embedded platforms becomes the next step in PdM. However, embedded platforms with limited resources require more careful design, and therefore which features to use to reduce the overall model complexity becomes crucial. Due to this reason, FR helps improve system design by offering perspective on the optimal number of features and from which part of the design they originate. Some of the existing approaches know the noisy features beforehand whereas others have specific metrics for evaluating particular kinds of FR methods. In addition to that, a manual search is expensive and oftentimes suboptimal and laborious.

The scope of the thesis is in the establishment of a methodology to compare different FR algorithms that allow to select features for optimized system performance when deployed on an embedded system. The overall information processing pipeline in PdM applications is narrowed down to state-of-the-art feature extractors, state-of-the-art FR algorithms, and finally an Machine Learning (ML) model is used as a wrapper only as a tool for comparison. However, finding the best solution for the PdM problem is beyond the scope of the thesis.

The methodology is tested on two datasets, PRONOSTIA which is an open-source machine health diagnostics dataset on finding bearing health, and a computer fan dataset which has been prepared at the IMMS GmbH on the health states of computer fans. Details of the design of the FR algorithms are addressed. The results of four different kinds of FR algorithms are compared using the proposed methodology. Finally, it is shown that the methodology offers a more detailed insight into which features aid to performance through this heuristic approach. This results in a considerable reduction in the number of features that would offer a better strategy towards system design for edge performance in PdM applications.

Kurzfassung

Das Feature Ranking (FR) befasst sich mit der Aufgabe, eine Hierarchie nützlicher Merkmale oder Features in einem gegebenen Datenset zu finden. Eine Anwendung von FR kann im Bereich der vorausschauenden Wartung (Predictive Maintenance, PdM) liegen, wo der Zustand der Maschine ständig überwacht wird. Da datengetriebene Ansätze immer beliebter werden, ist ein nächster Schritt hin zu PdM Anwendungen, Lösungen die auf künstlicher Intelligenz (KI) basieren auf eingebetteten Plattformen zu realisieren. Eingebettete Plattformen mit begrenzten Ressourcen erfordern jedoch ein sorgfältigeres Design der KI-Lösung, und daher ist es entscheidend, welche Merkmale verwendet werden, um die Gesamtkomplexität des Modells zu reduzieren. Aus diesem Grund hilft FR dem Systemdesigner, indem es eine Perspektive für die optimale Anzahl von Merkmalen und deren Herkunft bietet. Einige der bestehenden Ansätze kennen die verrauschten Merkmale bereits im Voraus, während andere spezifische Metriken für die Bewertung bestimmter FR-Methoden haben. Darüber hinaus ist eine manuelle Suche teuer und oft suboptimal.

Der Schwerpunkt dieser Arbeit liegt auf der Entwicklung einer Methodik zum Vergleich verschiedener FR-Algorithmen, die es ermöglichen, Merkmale für eine optimierte Systemleistung auszuwählen, wenn sie in einem eingebetteten System eingesetzt werden. Die gesamte Informationsverarbeitungspipeline in PdM-Anwendungen wird auf modernste Merkmalsextraktoren und modernste FR-Algorithmen eingegrenzt, und schlieich wird ein Modell des maschinellen Lernens (ML) als Wrapper als Vergleichswerkzeug verwendet. Die beste Lösung für die PdM-Aufgabe zu finden, würde jedoch den Rahmen dieser Arbeit sprengen.

Die Methode wird an zwei Datensätzen getestet: PRONOSTIA, einem Open-Source-Datensatz zur Ermittlung des Zustands von Lagern, und einem FAN-Datensatz, der von der IMMS GmbH über den Gesundheitszustand von Computerlüftern erstellt wurde. Es wird auf die Details des Designs der FR-Algorithmen eingegangen. Die Ergebnisse von vier verschiedenen Arten von FR-Algorithmen werden anhand der vorgeschlagenen Methode verglichen. Schlieich wird gezeigt, dass die Methode einen detaillierteren Einblick in die Frage bietet, wie viele Merkmale zur Leistung einer KI-Lösung beitragen und welche Merkmale durch diesen heuristischen Ansatz verrauscht würden. Dies führt zu einer beträchtlichen Verringerung der Anzahl von Merkmalen, die eine bessere Strategie für das Systemdesign für die Randleistung in PdM-Anwendungen bieten würde.

Contents

Chapter 1

Introduction

Artificial Intelligence (AI) is a trending topic and is applied to various applications. Of special interest are Predictive Maintenance (PdM) applications where a variety of signals and data is analyzed to derive a system state. With the advancement of embedded platforms in conjunction with the unprecedented success of data-driven methods, AI-based solutions bring embedded platforms to the next step in PdM. This is because it is often difficult to find deterministic solutions for a machine state. Furthermore, bringing solutions close to the machine reduces the cost of external processing and data transfer. However, embedded platforms with limited resources require more careful system design, and thereby a reduction of the model complexity and the pre-processing effort through finding the optimal set of features is of vital importance. Such scenarios with multiple possible sensors result in a challenge to select the best features of the machine learning task at hand. The selection should be based on the signal characteristics containing most information and contribute much to the analysis. Large design space is achieved from a wide array of sensors, sensor channels, spectrum information, etc, which makes manual selection unfeasible. Therefore, a Feature Ranking (FR) module becomes a requirement to obtain an overall lower model complexity, reduce pre-processing and achieve faster training time. Additionally, FR also helps in finding the optimum number of features to improve the performance of the model by choosing the right subset and preventing it from the Curse of Dimensionality problem, and also reducing the risk of overfitting.

To remain operational, industrial companies must keep their production means in good operating conditions by improving reliability and reducing maintenance costs. One of the possible strategies to solve this problem is the implementation of PdM by anticipating the occurrence of failure. The information processing pipeline is illustrated in Figure 1.1 which highlights the entire Machine Learning (ML) chain from raw data to model validation. Generally, the raw data is collected and cleaned with respect to the design of the problem at hand. Further, a pre-processing layer is applied to the cleaned data to extract useful information out of raw data and the optimal features are selected according to an FR

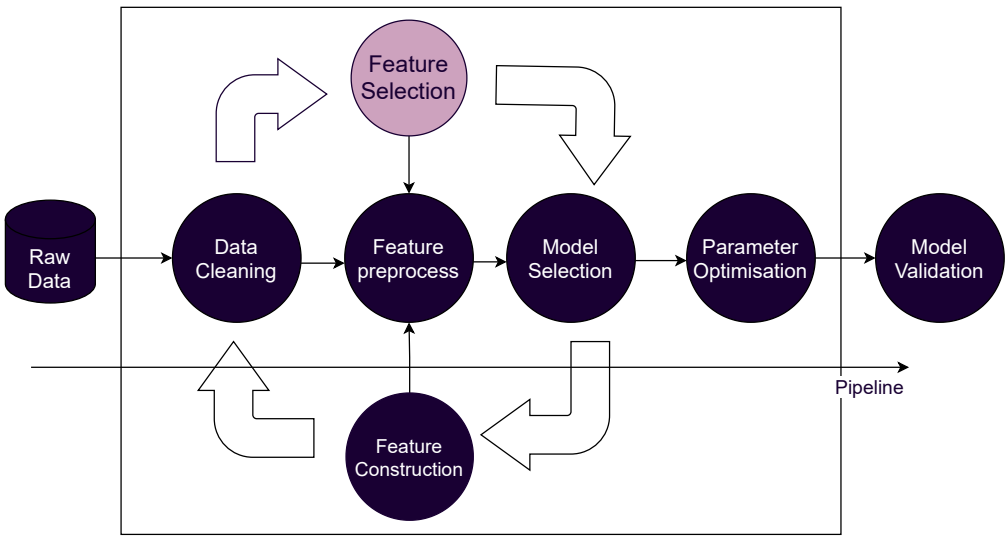


Figure 1.1 – Information processing pipeline

technique, this could also generate additional complex feature construction techniques like adding more feature extractors or extracting features out of augmented data. After the pre-processing layer, the features are then pushed into a model choosing some specific hyperparameters of the model, and the parameters are optimized through training and model validation according to a specific use-case. More generally a PdM system is an integration of six layers: sensors, signal processing techniques, condition monitoring (or fault detection), health assessment (or fault diagnostics), predictive decision support, and finally presentation layers. In order to explore the complexities involved in the pre-processing layer, a tool to compare extracted features and FR algorithms is required.

The primary research goal of the thesis is to evaluate FR methods to automatically select useful features and how to compare them. For a dataset with d features, if we apply a brute-force (trial and error method) approach with all possible combinations of features then total (2^d *-1*) models need to be evaluated for a significant set of features. It is a very time-consuming and labor-intensiv approach, therefore, we use feature selection techniques to find out the optimal set of features more efficiently and effectively. This particular problem motivates the requirement of an autonomous FR technique.

The motivation behind Feature Selection (FS) as illustrated in Figure 1.2 is to automatically choose a subset of features that is most relevant to the problem and thereby reducing model size, reducing training time, improving execution time, offering a framework to have a hierarchy of features embedded in the deployed model. In conjunction with PdM data, the number of sensors for each monitored device may be large, it is likely that the sensor values will be correlated and therefore redundant in model performance. The goal of FS is two-fold: an improvement in computational efficiency and reducing the generalization error of the model by removing irrelevant features or noise. Furthermore, most of the existing

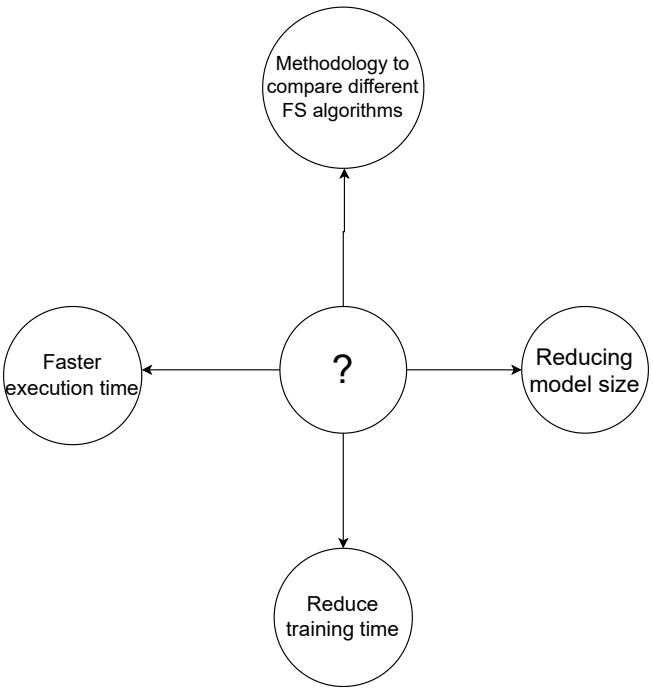


Figure 1.2 – Motivation behind a FR module for PdM applications

methods of comparison are restricted within a certain type of FR methodology, and thereby a tool for comparing multiple different methods is important. In addition, many existing methods explore scenarios where noisy features are known beforehand and that makes the testing of the performance of an algorithm straightforward. However, this is not available in real-world scenarios, and thereby a tool for a fair comparison is vital for the design of systems.

The scope of the thesis ranges from an investigation of the state-of-the-art (SOTA) FR methods motivated by the themes of the existing problems as described in Figure 1.2. Furthermore, developing a framework for a fair comparison of different types of FR methods would be evaluated by investigating the change in model performance with the number of features. However, it is important to convey that optimality of data cleaning along with feature extraction and fine-tuning of the AI model is beyond the scope of the thesis. The rest of the thesis is organized as follows. First, the existing approaches for FR and the fundamental tools necessary for the investigation of the thesis are discussed in Chapters 2 and 3. In Chapter 4, a detailed description of the datasets is provided mainly addressing the metadata information of the collected data. In Chapter 5, a discussion on the experimental design of the FR algorithms in conjunction with the context of the datasets is conducted. Chapter 6 discusses the comparison of the FR algorithms in terms of timing performance and performance metrics on which the FR algorithms are evaluated. Finally, in Chapter 7, the thesis is summarised and the scope of the future work is highlighted.

Chapter 2

State of the art

The basic principles behind FS and the information processing pipeline are covered briefly, setting a starting point for the remainder of this work. The questions addressed and researched during the conduction of this thesis are elaborated with the corresponding early work conducted in alignment with the research goal of this thesis :

- What kind of data-driven methods are applicable for FS in Predictive maintenance applications?[1]
- How could the dimensionality of the data be reduced without conducting model training?[13]
- Could a model offer interpretability, size reduction, and FS for one-time training?[16, 35]
- What are the tradeoffs involved in the timing and model performance for a given set of FS algorithms?
- How can a comparison be performed amongst the methods in the given methodology?[17]

Data-driven methods have gained popularity over the last decade with the advent of the development of machine-learning tools and frameworks, thereby making the data-driven methods for machine prognostics viable[1]. Statistical analysis is the fundamental building block of studying large amounts of data to generate models of predictive value. Statistical problem solving is achieved through a life-cycle as described in Figure 2.1 in four stages: a well-stipulated practical problem formulation, conversion of the requirements of the practical problem into a statistical problem, perform statistical analysis to obtain solutions and finally deploy it into practical solutions and update through a careful evaluation according to certain well-defined metrics. A detailed overview of fault diagnostics in conjunction with statistical signal processing methods are elaborated in [57]. Furthermore, in conjunction with PdM applications, the life-cycle as shown in Figure 2.2, the data is firstly gathered

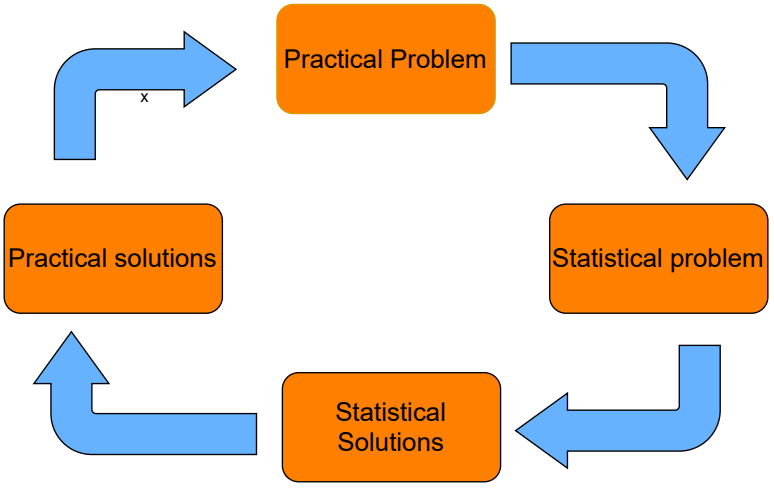


Figure 2.1 – Life-cycle for statistical problem solving

from as many different conditions representing the healthy and faulty operation. Then a pre-processing step is added in which FR comes as an important module. Further, a condition indicator is identified which distinguishes the healthy from the faulty data using a better set of features, this can be a threshold of certain statistical data or a certain specified threshold in the frequency of operation. The Remaining Useful Life (RUL) of the machine which is the length of time a machine is likely to operate before it requires repair or replacement is often used as a condition indicator[57]. This step is use-case specific, the condition indicators are often unknown and are hand-crafted according to a given scenario. Then a model is trained which finds out the relationship between the extracted features and the degradation status of the machine under investigation. Finally, the trained model can be deployed either on an edge device or in the cloud depending on the nature of the system design.

Figure 2.2 – Life-cycle for PdM applications[57]

The current SOTA algorithms investigated for FR in the thesis are suitable for offering solutions for PdM applications to some extent by providing specific metrics of feature importance. In addition to that, every PdM problem comprises of predicting the state of the machine from acquired data and thereby making multiple different FR methodologies viable for this application. However, there is a limitation in the comparison of algorithms as they are often compared with a certain metric applicable for a particular set of algorithms. In [17], the FS methods are compared based purely based on statistical indices like correlation coefficients, and in one of the scenarios investigated the noisy features are introduced as a linear combination of existing features, and a gaussian distributed noise is added further to

model the noisy features. However, in several real-world scenarios, the features could be correlated without a linear dependency. Therefore, an important contribution of the thesis is to establish a methodology where different variations of FR algorithms could be compared with a common metric and an investigation is conducted using datasets that are prepared inspired by real-world scenarios. This makes the problem challenging in two ways: a need to establish a common metric for evaluating the different FR algorithms and to tailor the FR algorithms for a deployment system in PdM applications. The further inquiries described above are investigated in detail in this chapter and the rest of the thesis.

The majority of the governing ML paradigm used for solving our PdM data-driven problem is Supervised Learning (SL). The defining characteristic of SL is the availability of annotated training data[56]. Typically, these labels are class labels or numerical values. SL algorithms induce models from these training data and these models can be used to classify other unlabelled data. In [35], the discussion of FR algorithm is conducted using synthetic datasets where the noisy features are modeled with a Gaussian distributed noise and the useful features are known beforehand for one of the scenarios. However, this information is generally not available for real-world scenarios that are used in PdM applications. This makes data-driven solutions in PdM applications challenging as a heuristic approach needs to be taken in order to find a hierarchy of useful features. The details of the datasets are discussed in Chapter 4.

FS becomes prominent, especially in datasets with many features. It will eliminate unimportant features and improve the performance of the problem at hand. The datasets involved during the conduction of the thesis involve two popular categories of ML problems in the SL paradigm: regression and classification. A wide array of features could be extracted from such datasets representing time-series data from the broadband and the spectral band and thereby making FR viable for the scenarios investigated. Essentially, FS becomes a dimensionality reduction technique, which tries to choose a small subset of the useful features from the original features by removing irrelevant, redundant, or noisy features, thereby making the data useful for analysis and future prediction. The thesis claims that FR for supervised machine learning tasks can be accomplished on the basis of the proposed methodology that is being developed where a selected subset of features yields predictive value with a certain class or label using certain kinds of FR algorithms. FR techniques can be categorized into three major approaches: 1. Filter Methods 2. Wrapper methods. 3. Embedded Methods. [13, 18, 20]

Feature Ranking Methods

2.1 Filter Methods

Filter methods perform a selection of features that are independent of any predictors, filtering out features that have little chance to be useful in analysis of data. The features are selected on the basis of their scores in various statistical tests for their correlation with the outcome variable[12]. Since there is no training involved, data-distribution statistics are generally used to evaluate a filter method. It is also called a univariate method, where the term variable is used synonymously as a feature. The simplest instance of the filter method used for feature importance is by investigating a simple Correlation matrix. The entries in the Correlation matrix are a metric of evaluating how correlated two features are. Unsupervised Learning is another ML paradigm where a type of algorithm learns from unlabelled data[13]. Another instance of this method is Principal Component Analysis (PCA), which is an unsupervised feature reduction method for projecting high-dimensional data into a new lower-dimensional representation of the data that explains or elaborates as much of the variance in the data as possible with minimum reconstruction error. Principal components are calculated using the Eigen Value Decomposition (EVD) of the data covariance matrix/correlation matrix or Singular Value Decomposition (SVD), usually after mean-centering the data for each attribute and giving a metric on the importance of features[13]. However, PCA doesn't explicitly rank each individual feature but ranks it collectively as a linear combination of features.

There has been considerable interest in the use of Principal Component Analysis, as a means of reducing the effective measurement space. The beginnings of PCA are found in the works of Karl Pearson (1901). The PCA algorithm can be summarised as follows[13]:

1. The n observations for each of m features from sensors is collected into m raw data vectors, x. The data x is then standardised by mean centering according to Equation (2.1) where μ and σ are the mean and standard deviation along the axis of features and i and j represents the row and column indices of the data matrix of size nxm :

$$\tilde{x_{ij}} = \frac{x_{ij} - \mu}{\sigma} \tag{2.1}$$

2. The standardised data $\tilde{x_{ij}}$ describing the feature space are stored in a (nxm) matrix $Y = [\tilde{x_{ij}}]$, with m features for each observation arranged on each of n rows. The resultant standardised data matrix is denoted as Y . In addition, we initialize a matrix holding the current representation of the data as modelled by PCA. $X = 0$ and the principal component counter : $k = 1$.

3. The correlation matrix, A, which is symmetrical and positive definite, is formed according to the formula: $A = Y^T.Y$ with a dimensionality (mxm). The first (largest) eigenvalue of A, λ_k and its corresponding eigenvector, p_k are computed for this matrix; this eigen vector is the next most significant principal component basis function.

   An interesting property of principal components is that they are rotation invariant. This makes it more robust as a normal correlation matrix as correlation coefficients change with a rotational change in the data vectors $\tilde{x_{ij}}$. The coefficients of the largest principal component corresponding to the feature importance of a particular feature. In practice, however, mostly 95% of the variance is generally considered for evaluating feature ranking. One major advantage of PCA is that it is Unsupervised and requires no label and is therefore identical for both regression and classification scenarios.

Another popular filter method is the Analysis of Variance (ANOVA) used in finding if multiple groups of samples come from the same distribution. A group here is generally referred to as an input categorical variable. Usually for this test, a null hypothesis (H_0) is formed which states that they come from the same distribution and an alternative hypothesis (H_1) is formed that states that they are not from the same distribution. The hypotheses are then tested (specified here for $k = 6$ groups) about the sample means (μ_j)[11] :

$$H_0 : \mu_1 = \mu_2 = \mu_3 = \mu_4 = \mu_5 = \mu_6$$
$$H_1 : \text{ Not all } \mu_j \text{ are equal } (j = 1 : 6)$$

However, in conjunction with FR, this is not an appropriate method as most of the input features in PdM applications are continuous in nature. However, it can be useful for targeting a problem with a different objective of finding useful Health Indicators (HI)[10].

2.2 Wrapper Methods

Wrapper methods are a class of FR algorithms that often give better results (in terms of final predictive performance of a learning algorithm) than filters because FS is optimized for the particular learning algorithm used. However, since a learning algorithm is employed to evaluate every set of features considered, wrappers are prohibitively expensive to run in terms of their high training times and can be intractable for large databases containing too many features[14].

Filter methods however can execute many times faster than wrappers, and therefore stand a better chance of scaling to databases with a large number of features than wrapper do[18]. Furthermore, filters do not require re-execution for different learning algorithms. However, wrappers can handle better noise in the data[29].

Wrapper methods follow a greedy search approach by evaluating multiple possible combinations of features against a performance metric for evaluation. The performance metric depends on the type of problem, e.g. For regression evaluation criterion can be p-values, R-squared, Mean Squared Error (MSE), similarly for classification problems, the evaluation metric can be accuracy, precision, recall, f1-score, etc. depending on the skewness of the dataset. Finally, it selects the combination of features that gives the optimal results for the specified machine learning algorithm[2, 6, 51]. The most common flavors of wrapper methods are :

1. Sequential Forward Selection (SFS): This algorithm is a bottom-up search procedure which start from a null set and gradually adds features in a greedy fashion selected by some evaluation function[9]. At every stage, the estimator chooses the best feature to add or remove based on the cross-validation score of an estimator.

Algorithm 1 Sequential Forward Selection (SFS)

Require: $Y = \{\Phi\}, k = 0$
Ensure: $J(x) \leftarrow$ Cost function for evaluation
  $Y \leftarrow$ Set of optimal features
  $k \leftarrow 0$ ▷ This is a counter
  $N \leftarrow$ Total number of features to be selected
  while $k \neq N$ **do**
  $\hat{x} = \arg\min_{x \in Y_k} J(Y_k + x)$ ▷ Select the next best feature
  $Y_{k+1} = Y_k + \hat{x}$ ▷ Update the feature set and the counter
  k =k+1

  end while

Figure 2.3 – Sequential Forward Selection

2. Sequential Backward Selection (SBS): This algorithm is very similar to SFS with the difference that the selector removes features to form a feature subset.

Algorithm 2 Sequential Backward Selection (SBS)

Require: $Y = X, k = 0$
Ensure: $J(x) \leftarrow$ Cost function for evaluation
  $Y \leftarrow$ Set of optimal features
  $k \leftarrow 0$ ▷ This is a counter
  $X \leftarrow$ Complete feature set
  $N \leftarrow$ Total number of features to be selected
  while $k \neq N$ **do**
  $\hat{x} = \arg\min_{x \in Y_k} J(Y_k - x)$ ▷ Remove the worst feature
  $Y_{k+1} = Y_k - \hat{x}$ ▷ Update the feature set and the counter
  k =k+1

  end while

Figure 2.4 – Sequential Backward Selection

3. Recurcive Feature Elimination (RFE): The flowchart of this algorithm is described in Figure 2.5[55]. There are two important configuration options when using RFE: the choice in the desired number of features to select and the choice of the algorithm used to help choose features. RFE works by searching for a subset of features by starting with all features in the training dataset and iteratively removing features until the desired number remains.

   This is achieved by fitting the given ML algorithm used in the core of the model, ranking features by importance, discarding the least important features, and re-fitting the model. This process is repeated until a specified number of features remains[15].

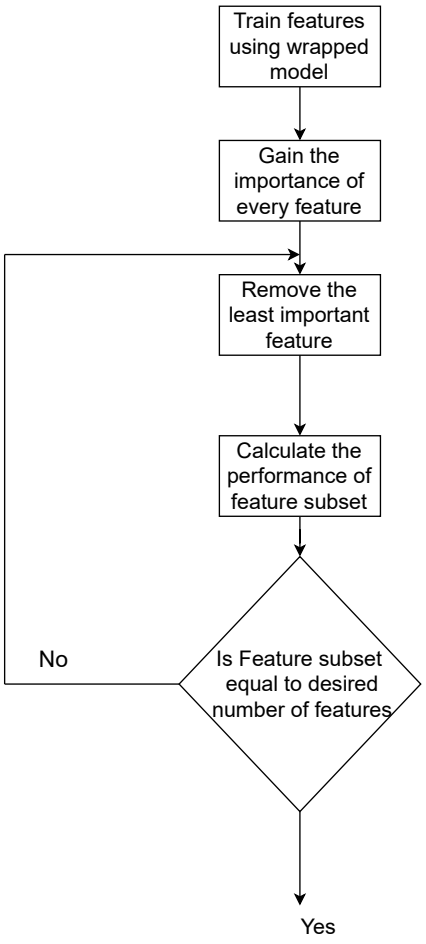


Figure 2.5 – Recursive Feature Elimination[55]

2.3 Embedded Methods

For Embedded methods, the FR is an integrated part of the algorithm. Embedded methods combine the qualities of filter and wrapper methods. It's implemented by algorithms that

have their own FS methods in them. This is oftentimes achieved through offering a penalty term that regularises the coefficients of feature weights which are directly interpreted as feature importance. Furthermore, with the advent of Deep Learning (DL) in the past decade along with the availability of faster computational resources, a wide array of DL based approaches have been helpful in the area of FS as well as offering interpretability to the models[16, 35, 53].

There are a wide array of Embedded methods as mentioned in [16, 25, 53], however, two different types of Embedded methods are carried out as a part of the thesis, where one is a linear model called *Linear Absolute Shrinkage and Selection Operator* (LASSO) [26] and a promising technique proposed in [16] by adding an extra layer at any *Deep Neural network* called as *CancelOut*. These methods are chosen because of the simplicity of their design and offer good performance in a plethora of applications like Computer Vision, Medical Imaging, *e.t.c.* where dimensionality reduction is a prominent requirement[29]. This subsection explores the nature of these methods and would be implemented for the PdM use-case further in the thesis.

The advantages of Embedded methods are that they take into consideration the interaction :

1. They have one-time training involved. This is because the FR module is embedded within the ML model and thereby results in performing FR and model training simultaneously in a single training as opposed to multiple trainings offered by wrappers.

2. They are oftentimes as fast as filter methods with better performance. The improvement in performance is offered by the addition of a model that is absent in filter methods with a trade-off in performance.

3. They are much less prone to over-fitting. This is due to the fact that only a subset of features are considered at a time due to regularisation penalty.

However, the disadvantages of Embedded methods are oftentimes in the optimal search of the hyperparameters involved in regularization. In addition to that, the choice of model that performs the solution of the problem can be a computationally expensive operation.

2.3.1 LASSO

In recent years, the problem of finding solutions of under-determined systems of linear equations attracted enormous attention[24]. Given a substantial interest in the problem (and especially that it is coming from a wide array of different fields) in conjunction with solving a variety of current problems, it can expand further in the range of its applications. One such approach to solve an under-determined system of equations is *Linear Absolute Shrinkage and Selection Operator* (LASSO) was first formulated by Robert Tibshirani in

1996[26]. There are many variants of LASSO but the following one is probably the most well known in [27]:

$$\hat{\boldsymbol{x}} = \min_{x} \frac{1}{2} \|\boldsymbol{y} - \boldsymbol{A} \cdot \boldsymbol{x}\|_2^2 + \lambda \cdot \|\boldsymbol{x}\|_1 \tag{2.2}$$

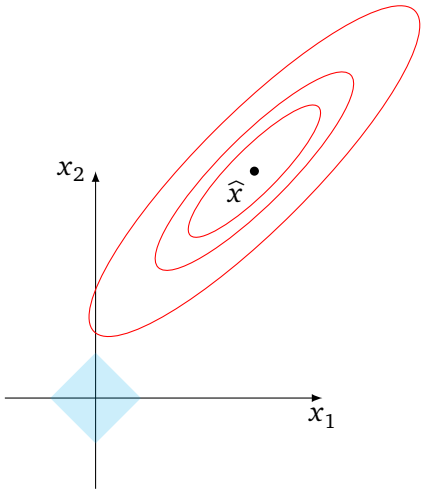


Figure 2.6 – LASSO Regression[58]

for $\boldsymbol{x} \in \mathbb{R}^M$, $\boldsymbol{A} \in \mathbb{R}^{M \times N}$, and $\boldsymbol{y} \in \mathbb{R}^N$ and λ denotes the Lagrangian multiplier (and generally $M \gg N$). The solution to Section 2.3.1 is visualised in Figure 2.6, where x is the vector representing the features, A represents all the data collected with M measurements and N is the total number of features. The solution to the LASSO problem is the joint optimisation both the $l2$ norm of the error which is the first term in Section 2.3.1 as visualised in red in Figure 2.6 and the $l1$ norm of x as visualised in blue. The " p-norm" of a vector x with i components is denoted as $\|x\|_p$. A p-norm is defined in Equation (2.3) as in [28]:

$$\|x\|_p = \left(\sum_i |x_i|^p \right)^{1/p} \tag{2.3}$$

The $l1$ norm $\|\boldsymbol{x}\|_1$ is a diamond shaped structure as visualised in Figure 2.6 and is our regularisation factor in LASSO, it is due to the sharp edges of the diamond, a dimensionality reduction is achieved through offering sparsity in the solution of x as both the error and the regularisation term has to intersect simultaneously to form the optimal solution and thereby achieving a feature rank which are the coeffients of $\hat{\boldsymbol{x}}$.

In other words, LASSO puts a constraint on the sum of absolute values of the model parameters, the sum has to be less than a fixed value (upper bound). In order to do so the method apply a shrinking (regularisation) process where it penalizes the coefficients of the regression variables shrinking some of them to zero to achieve FS. During FS process the

variables that still have a non-zero (both positive and negative) coefficient after the shrinking process are selected to be a part of the model. The goal of this process is to minimize the prediction error. In practice the tuning parameter λ (regularization coefficient), that controls the strength of the regularization penalty, assume a great importance in terms of tuning. This is because it is related to the *sparsity* of the obtained solution. Sparsity is defined as the $l0$ norm of a vector, which in other words is the number of non-zero elements in a vector[27]. The greater the value of λ, the more number of noisy features are thresholded to zero. Indeed when λ is sufficiently large then coefficients are forced to be exactly equal to zero, this way the *dimensionality* can be reduced[58]. The larger this parameter λ , the more number of features are not taken into consideration. On the other hand, if $\lambda = 0$ we have OLS (Ordinary Least Squares) regression. Finally, LASSO can be used in PdM applications as it helps to reject the noisy features leaving only a subset of useful features for investigation[20].

2.3.2 CancelOut

One of the effective ML methodologies is DL which can approximate any compactly supported continuous function on $\mathbb{R}^N$ with a single hidden layer feed-forward neural network (NN). Because of the complex nature of Artificial Neural Networks (ANNs), DL models are mostly treated as black boxes. In [31], linear models with elastic-net regularisation are used in conjunction with a NN with multiple layers.

In this subsection, a brief review of CancelOut (CO) which is a new layer for DNNs, and its functionality on how to help identify a subset of relevant input features (variables) in a dataset is discussed[16].

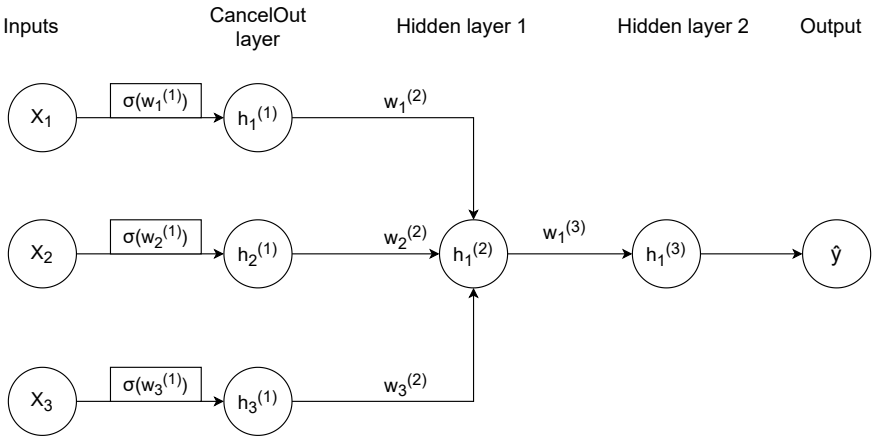


Figure 2.7 – A DNN with CancelOut in the preprocessing layer

CancelOut is an ANN layer where neurons in the CancelOut layer have only one connection to one particular input unlike a fully connected (FC) layer (Figure 2.7). The fundamental

idea of this method is to update its weights (W_{CO}) during the training stage so that *noisy* features are automatically $canceledout$.

$$CancelOut(X) = X \circ g(W_{CO}) \tag{2.4}$$

where $\circ$ denotes an element-wise multiplication or a hadamard product, $\mathbf{X}$ is an input vector $\mathbf{X} \in \mathbb{R}^N$ (all the features), W_{CO} is a weight vector $W_{CO} \in \mathbb{R}^N$ (the CancelOut layer), N is the feature size , and g is an activation function, where $g(x)$ denotes here element-wise application , e.g. $\boldsymbol{X} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$, then $g(\boldsymbol{X}) = g\left(\begin{bmatrix} a \\ b \\ c \end{bmatrix}\right) = \left(\begin{bmatrix} g(a) \\ g(b) \\ g(c) \end{bmatrix}\right)$

A more detailed theoretical analysis of $CancelOut$ is available in [16]. It is, however, important to discuss the hyper-parameters of $CancelOut$.

$$L(X,Y) = \mathscr{L}(X,Y) - \lambda_1 \operatorname{var}\left(\frac{W_{CO}}{N_v}\right) + \lambda_2 \left\| \frac{W_{CO}}{N_v} \right\|_1$$

where $L(X,Y)$ is the overall loss function after the integration of a cancelout layer and $\mathscr{L}$ is a selected loss function for a particular problem. For the respective regression or classification task, λ_1 and λ_2 are user-specified coefficients $\lambda_1 \in [0,1]]$, $\lambda_2 \in [0,1]$, $N_v$is a number of variables in a dataset, and W_{CO} are $CancelOut$ weights.

The variance of the weights from the CancelOut layer var $\left(\frac{W_{CO}}{N_v}\right)$ helps to improve diversity in the CO layer, there $l1$ norm is used to introduce sparsity in W_{CO} weights and to constrain the variation to small weights. Also, $l1$ penalty restricts the model from selecting correlated features. Furthermore, *negative CO weights are interpreted as feature rejection*. Lastly, $Cancelout$ supports all losses and does not require the realization terms.

$CancelOut$ method is efficient and it can be used in all modern DL frameworks along with mainly *SL* use cases. Finally, this method helps understand the dependency of the data and its influence on the performance of DL models.

The following three types of FR techniques are investigated in the next chapters with two different datasets with an attempt to explore the research questions as illustrated at the beginning of this chapter. Since, FR is still an active research area because of the requirements coming from data storage, model performance, and deployment, a strong focus shall be given in this thesis investigating the trade-offs of the different approaches in finally achieving optimality for deployment system performance.

Chapter 3

Fundamentals

This section reviews the details of the fundamentals involved to frame the research questions for the thesis. It involves a description of the background of the feature extractors and the ML paradigms that will be investigated later.

3.1 Feature Extractors

Feature extraction is a crucial part of the pre-processing of signals as it helps in performing compression by extracting meaningful information out of raw data. There are broadly two ways to extract features from time-series data: broadband and narrowband (spectral information) approach. It is important that the spectral parameters are narrowband so that useful information could be extracted from each part of the frequency spectrum. For this thesis, four kinds of feature extractors were investigated for the scenarios involved during its conduction. The details of the feature extractors are mentioned below keeping in focus how they can be applied in raw time-series data. These extractors however could be used for a wide array of other use cases. However, the parametric design of the feature extractors would be restricted to a particular use case. There are three degrees of freedom for extracting features: accelerometer axes (space), time-series data from these axes (yielding to broadband features), and changing this data into a time-frequency domain for narrowband analysis (yielding to spectral features).

3.1.1 Broadband Statistics

Usually, for analyzing time-series data, signal-based statistical metrics that can be applied to any kind of signal are useful as changes in these features can indicate changes in the health status of the system. Moments are a set of defined parameters that describes a statistical distribution. The basic statistics involve mean, variance and skewness, whereas, higher-order statistics provide insight into system behavior through the fourth moment (kurtosis) and

the third moment (skewness) of the vibration signal. All these statistics can be expected to change as a deteriorating fault signature intrudes upon the system under test. There are 6 statistical features that are used in analyzing the raw data in the broadband[52].

- RMS: For a given dataset, the arithmetic mean (μ), is the average of a set of N observations as described in Equation (3.1) as the sum of all the observations divided by the total number of observations.

$$\mu = \bar{w} = \frac{1}{N}\left(\sum_{i=1}^{N} w_i\right) = \frac{w_1 + w_2 + \cdots + w_N}{N} \tag{3.1}$$

  RMS is defined as the square root of the mean square (the arithmetic mean of the squares of a set of N observations). This variable represents a measure of total power in a signal. It is described by the formula below in Equation (3.2):

$$x_{\text{RMS}} = \sqrt{\frac{1}{N}\left(x_1^2 + x_2^2 + \cdots + x_N^2\right)} \tag{3.2}$$

- Variance: The variance of a random variable W is the expected value of the squared deviation from the mean (μ) of $W, \mu = \text{E}[W]$ (where E[.] is the Expectation operator) :

$$\text{Var}(W) = \text{E}\left[(W-\mu)^2\right] \tag{3.3}$$

  The variance of a collection of N equally likely values can be written as described in Equation (3.4) as:

$$\text{Var}(W) = \frac{1}{N}\sum_{i=1}^{N}(w_i - \mu)^2 \tag{3.4}$$

  The variance is a measure of dispersion, which illustrates how far a set of observations is spread out from their mean value. This is also the energy of the signal.

- Kurtosis: The Kurtosis symbolises how much the signal is prone to outliers. Developing faults or defects in machinery could increase the number of outliers, and consequently increase the value of this metric. The kurtosis has a value of 3 for a Normal distribution (Gaussian distribution). The kurtosis is the fourth standardized moment, defined as:

$$\text{Kurt}[W] = \text{E}\left[\left(\frac{W-\mu}{\sigma}\right)^4\right] = \frac{\text{E}\left[(W-\mu)^4\right]}{\left(\text{E}\left[(W-\mu)^2\right]\right)^2} = \frac{\mu_4}{\sigma^4}, \tag{3.5}$$

  where μ_4 is the fourth central moment and σ is the standard deviation.

- Skewness: Skewness is a measure of asymmetry of a statistical distribution. The Skewness can be positive, negative, zero or undefined. Faults in a machinery can

reflect on the distribution symmetry and therefore increase the level of skewness. The skewness also defined as the third moment ($\tilde{\mu}_3$)is described as:

$$\begin{aligned}\tilde{\mu}_3 &= \mathrm{E}\left[\left(\frac{W-\mu}{\sigma}\right)^3\right] \\ &= \frac{\mathrm{E}\left[W^3\right]-3\mu\mathrm{E}\left[W^2\right]+3\mu^2\mathrm{E}[W]-\mu^3}{\sigma^3} \\ &= \frac{\mathrm{E}\left[W^3\right]-3\mu\left(\mathrm{E}\left[W^2\right]-\mu\mathrm{E}[W]\right)-\mu^3}{\sigma^3} \\ &= \frac{\mathrm{E}\left[W^3\right]-3\mu\sigma^2-\mu^3}{\sigma^3}\end{aligned} \tag{3.6}$$

- Shape factor: Shape Factor (SF) is the Root Mean Square (RMS) divided by the mean of the absolute value of a signal. Shape factor depends on the signal shape and does not depend of the signal dimensionsality.

$$x_{SF} = \frac{x_{rms}}{\frac{1}{N}\sum_{i=1}^{N}|x_i|} \tag{3.7}$$

- Crest factor: Crest Factor (CF) is the peak value divided by the RMS. Faults often could be detected by the alterations in the peaks of a given signal before they are identified by the energy (variance) of a signal. Therefore, it could provide an early warning for faults when they begin to develop.

$$C = \frac{\left|w_{\text{peak}}\right|}{w_{\text{rms}}} = \frac{\|w\|_\infty}{\|w\|_2} \tag{3.8}$$

Spectral features

To further the investigation of feature extraction, it is useful to explore the dataset while transforming it into a frequency and time-frequency domain to extract more essential features of the vibration data that might help further the diagnosis of this problem of early fault detection. Out of several existing possibilities, the thesis restricts to three feature extractors in this category which are widely used for PdM applications.

3.1.2 Discrete Wavelet Transform

The Discrete Wavelet Transform (DWT) is of practical use in signal processing applications[32]. They help achieve significant compression and denoising by transforming the data in the time-frequency domain. The Fourier Transform (FT) gives the spectral components in the signal. However, it does not provide any specification of the time-localization of the spectral

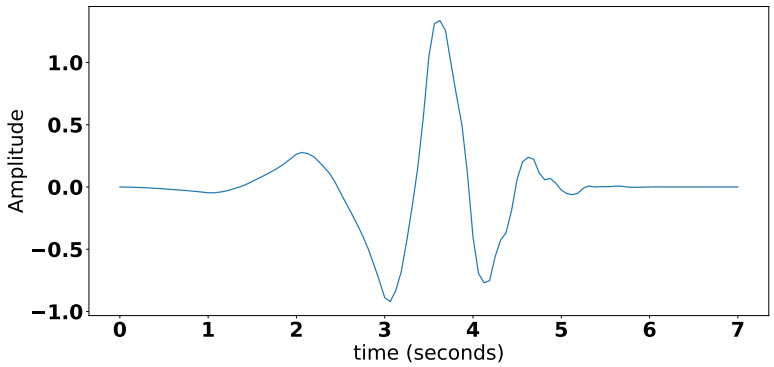
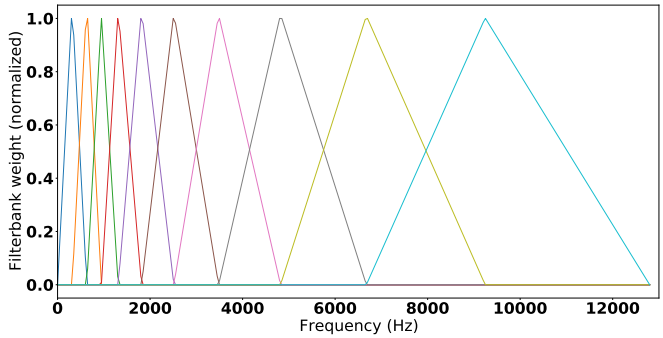


Figure 3.1 – Daubechies (db4) wavelet[32]

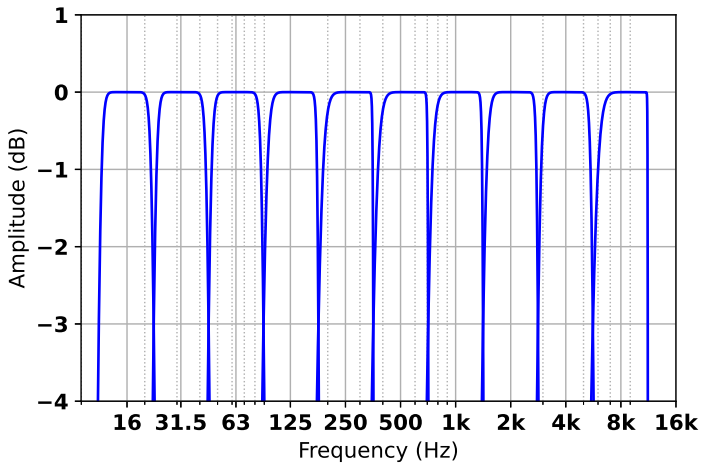
components. Therefore, a time-frequency representation of the signal is needed especially for PdM applications where events representing a healthy and a faulty state are localized in time as oftentimes a particular spectral component occurring at any instant can be of particular interest. On such occasions, it might be useful to know the time intervals where these particular spectral components occur. The Wavelet Transform is obtained by first passing the time-domain signal through various highpass (HP) and lowpass (LP) filters, which filters out the sections of the high and low-frequency regions of the spectrum and returns a time-domain signal in which half of the samples are discarded after filtering as per the Nyquist criterion.

The low pass of the signal is further iteratively filtered. The filters have a small number of coefficients and have good computational performance. Additionally, these filters can reconstruct the sub-bands while canceling any aliasing that occurs due to downsampling. For example, for a signal which has a Nyquist frequency of 1000 Hz and we want a 4 level decomposition of the signal, we first break the signal into 0-500Hz (LP portion), and 500-1000Hz (HP portion). Further, the LP portion is divided into 0-250 Hz (LP) and 250-500Hz (HP) and thereby obtaining 4 levels of decomposition. A Daubechies (db4) wavelet widely used for investigating signal discontinuities is shown in Figure 3.1 and used as a basis for the DWT. The DWT process is equivalent to decomposing a signal with discrete multi-rate analysis filterbanks.

3.1.3 Logarithmic Filter Bank



(a)



(b)

Figure 3.2 – (a) Mel-scaled Filter Bank (b)Octave Filter Bank Frequency Response

A Logarithmic Filter Bank (LFB) is a digital filter bank that is widely used in audio-signal processing applications as the human perception of loudness is calibrated logarithmically in the frequency domain for the audible range. This approach mitigates the shortcoming of the Short-Time Fourier Transform (STFT) by extracting features specific to different parts of the frequency spectrum and simultaneously achieving a dimensionality reduction. The features obtained from this approach are through a three-stage process: taking short chunks of the input waveform followed by obtaining a Power Spectral Density (PSD) of the individual chunks in the logarithmic scale and finally, a mel-scaled filter bank is applied to the PSD[4]. In Figure 3.2(a), a Mel-scaled Filter Bank for 10 frequency bins is visualized for a Nyquist frequency of 12.8KHz. It can be observed that the center-frequencies are spaced with higher resolution for lower frequencies and lower resolution for higher frequencies[34, 54]. A

dimensionality reduction is also achieved as the mel-scaling reduces the standard frequency domain resolution by prioritizing different parts of the spectrum. Additionally, one of the simplest and oldest ways to detect faults in machinery is by listening to the sound of the machine and tracking the abrupt changes in its behavior over time[50]. Due to the multiresolution property of the LFB, this becomes an interesting candidate for extracting features in vibration data.

3.1.4 Octave Filter Bank

The Octave Filter Bank (OFB) is an array of Infinite-duration Impulse Response (IIR) filters applied over the spectrum of a signal that decomposes a signal into octave subbands with variations in the frequency resolution[5]. An octave band is a frequency band where the highest frequency is twice the lowest frequency. In Figure 3.2(b), we can see an OFB with 10 bands varying from a center frequency of 16Hz to 16KHz. Here, the center frequencies are one octave apart from each other resulting in narrower bands in the lower parts of the spectrum and broader bands in the higher part of the spectrum. This kind of multi-resolution approach is oftentimes helpful as lower frequencies with more frequency resolution might offer helpful features from the signal and higher frequencies with better resolution in time offer better features in a general broadband analysis. The power in the octave band is taken as a feature for this feature set.

3.2 Supervised Learning

A SL strategy is adapted for the scenarios investigated in the thesis due to the scope and availability of labeled data. Therefore, the ML frameworks can be trained after the extraction of the features with a well-defined relationship that relates the features to the labels. The task of the ML algorithms is to establish a meaningful relationship that relates the features to the labels and thereby make a model of the given data at hand. The models investigated for the conduction of this thesis are of two categories: a linear model and a non-linear model. Further, all models must show certain generalizability to perform over unseen test data.

3.2.1 Linear model

A linear model aims to establish a linear relationship that fits the training data. There are two approaches conducted, one for solving *regression* and the other for solving *classification* problems. Therefore, primarily two models are investigated: linear regression and logistic regression. An array of hyper-parameters are also important for the design of these models for performing optimization.

Linear regression

Linear regression is perhaps one of the most popular and well-understood algorithms in statistics and ML. This is due to its ease of interpretability.

Given a data set $\{y_i, x_{i1}, \ldots, x_{im}\}_{i=1}^{m}$ of m examples, a linear regression model assumes that the relationship between the dependent variable y (label) and the m-vector of regressors $\mathbf{x}$ (features) is linear. This relationship is further modeled through an error variable ε— which relates to the difference from the outcome of the model (regressors) to the ground truth (the dependent variable). Thus the model takes the form as in [21]:

$$y_i = b_0 + b_1 x_{i1} + \cdots + b_m x_{im} + \varepsilon_i = \mathbf{x}_i^\top b + \varepsilon_i, \quad i = 1, \ldots, m$$

where $^\top$ denotes the transpose, so that $\mathbf{x}_i{}^\top b$ is the inner product between vectors $\mathbf{x}_i$ and b and ε represents the error . Often these n equations are stacked together and written as

$$\mathbf{y} = X b + \varepsilon$$

The difference of the ground truth of the dependent variable from the predicted output of the model is what needs to be reduced for optimizing the model. This is often referred to as the *cost function* or the *loss function*. However, there are varying flavors of the loss function and is dependent on the use case. A common loss function is a Mean-squared error (MSE) loss for regression problems, where the square of the differences between the predicted and the original variable is taken into consideration. The purpose of model training is to reduce this loss term by updating the model parameters. The training is conducted with the help of variations of the gradient descent algorithm which aims at reducing this loss term[38]. Therefore, a set of hyper-parameters that involve optimization are the optimizer, regularisation parameter by adding a $l1$ or a $l2$ penalty the loss function and the regularisation coefficient which is a measure of how much regularisation needs to be considered[39, 40].

Logistic regression

One known discriminative classifier is the logistic regression algorithm, proposed for machine learning architectures. It computes discrete outputs and specializes in classification tasks. This algorithm is used for both binary and multi-class classification problems. For the case of multiple classes, the algorithm can be adapted by using a "one vs rest" or a "one vs one" approach. The classifier uses the logistic/sigmoid function that is expressed as $f(z) = \frac{1}{1+e^{-z}}$ and can map real values into the domain $[0, 1]$. In ML, the predictions made by the model architecture and are mapped to probabilities.

$$P(y=1) = \frac{1}{1+e^{-(xw+b)}} \tag{3.9}$$

$$P(y=0) = 1 - P(y=1) = \frac{e^{-(xw+b)}}{1+e^{-(xw+b)}} \tag{3.10}$$

To illustrate with a binary-classification task, before placing the given input into the class, the probability of fitting in either one of the classes is calculated and can have any value between 0 or 1. In Equation (3.10) and Equation (3.9) as described in [22], the calculation process for the probabilities can be observed, where w are weights, b, bias values, x the input, and y the output. The weights and biases are the trainable parameters for the logistic-regression problem. To determine exactly the placement of the sample, a decision boundary is set prior to numerically separate the two classes. For a multi-class classification problem, the "one vs rest" approach involves splitting the multi-class dataset into multiple binary classification problems. A binary classifier is then trained on each binary classification problem and predictions are made using the model that is the most reliable. One of the popular loss functions for the multi-class classification problem is *Categorical-cross entropy*[22]. Additionally, regularisation parameters and the optimizer used is a hyper-parameter for this ML approach.

3.2.2 Multi-layer perceptron

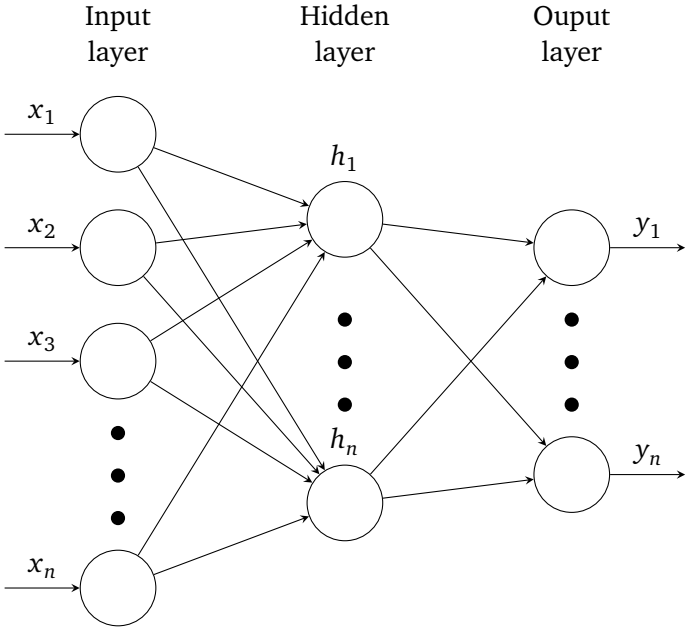


Figure 3.3 – Multi-layer Perceptron (MLP)

There is a wide array of non-linear models approached with the advent of DL. A very simple approach is involved during the conduction of the thesis with a *Multi-layer Perceptron (MLP)* as displayed in Figure 3.3. This survey is narrowed as it is only used as a tool to compare the algorithms under investigation. The MLP generally contains an input layer with a set of hidden layers and an output layer. Each layer contains "neurons" that have trainable parameters which are called weights (W) and biases (b).

The hidden and final layers are usually followed by activation functions. In Artificial Neural Networks (ANNs), the activation function of a node is defined as the output of that node given an input vector. There are various kinds of activation functions used in the MLP architecture[23]. One popular activation function is the *Rectified Linear Unit* (ReLU). Major benefits of ReLUs are sparsity and a reduced likelihood of vanishing gradient. The definition of a ReLU (represented by h) is as in [8]:

$$h = max(0, y) \tag{3.11}$$

where $y = W.X + b$, is a result at the output of each neuron and the max function takes the maximum of the two input parameters. One major benefit of using this activation function is the reduced likelihood of the gradient vanishing. This arises when y is positive. In this regime, the gradient has a constant value. In contrast, the gradient of sigmoids (similar activation as the logistic function) becomes increasingly small as the absolute value of x increases. The constant gradient of ReLUs results in faster learning. Additionally, for negative values of y, sparsity in the MLP is achieved as that neuron gets deactivated resulting in a lesser number of trained parameters. The more such units that exist in a layer the more sparse the resulting representation. Sigmoids on the other hand are always likely to generate some non-zero value resulting in dense representations. Sparse representations could be more beneficial than dense representations as they lower the overall model size. For multi-class classification problems for M classes, at the final layer, a 'softmax' function as defined in 3.12 is used that gives the probabilities of occurrence of each class[7].

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{M} e^{z_j}} \quad \text{for } i = 1, \ldots, M \text{ and } \mathbf{z} = (z_1, \ldots, z_M) \in \mathbb{R}^M \tag{3.12}$$

where M is the number of total classes and z is the output at each neuron in the final layer.

The choice of the number of layers and the number of neurons in each layer is again a design problem that directly affects the edge performance. The MLP operates by fitting a non-linear curve to the available data at hand. By the *universal approximation theorem*[42], a DNN can make representations of data, however, a risk of its performance over unseen test data is a challenging problem in the design of these structures. This is a problem of *overfitting* and the design of hyper-parameters are crucial to mitigate this problem. A dropout layer is introduced usually as a regularisation technique and the dropout rate is a

hyperparameter. Large dropout rates regularize the network more and are often useful to overcome the task of overfitting[30]. The loss function is dependent on the task at hand. For classification tasks, a categorical cross-entropy loss function is one popular choice used and for regression problems, a Mean-squared loss is widely used[37]. A back-propagation algorithm is used in the training of MLPs. A wide array of optimizers are used which are variations of the stochastic gradient descent method (SGD) to train neural networks, out of which *Adam* and *RMSprop* are popular for this particular paradigm[36, 37].

The training is conducted in batches and one complete forward and backpropagation for all the batches is one epoch[2]. The MLP is a stochastic model and hence it often takes larger epochs for training them. Larger batches take smaller training time with the tradeoff being that the model might fail to learn useful representations of the data.

Chapter 4

Datasets

Datasets are a crucial element for developing an ML/DL framework since they provide a way of training the models and allow benchmark evaluation for different approaches. Large-scale datasets are of enormous importance for the success of deep learning, particularly for the end-to-end deployment of models, where training with large amounts of data shows promising results[43]. Despairingly, due to the confidentiality of sensitive production to machine processed data, these are not that easy to come by. Public data-sets in conjunction with PdM applications usually elude this by conducting experiments to gather large volumes of data under varying conditions from a plethora of use-cases that can be involved in the analysis for further Research and Development (R & D). The datasets that were investigated primarily for the investigation of FR algorithms for this thesis are from two different sources, the PRONOSTIA dataset, which is a public dataset, and another dataset investigating the health of computer fans prepared at the Institut für Mikroelektronik - und Mechatronik-Systeme gemeinnützige GmbH (IMMS).

4.1 PRONOSTIA

PRONOSTIA is an experimental platform that enables testing, verifying, and validating methods related to bearing fault detection, diagnostic and prognostic approaches. A more detailed description of how the data was collected is mentioned in [33]. The PRONOSTIA setup is described in Figure 4.1.

Bearing data representing 3 different loads were considered during the conduction of the measurements for this datastet as summarised as follows:

1. First operating Conditions: 1800 rpm and 4000N;
2. Second operating Conditions: 1650 rpm and 4200N;
3. Third operating Conditions: 1500 rpm and 5000N;

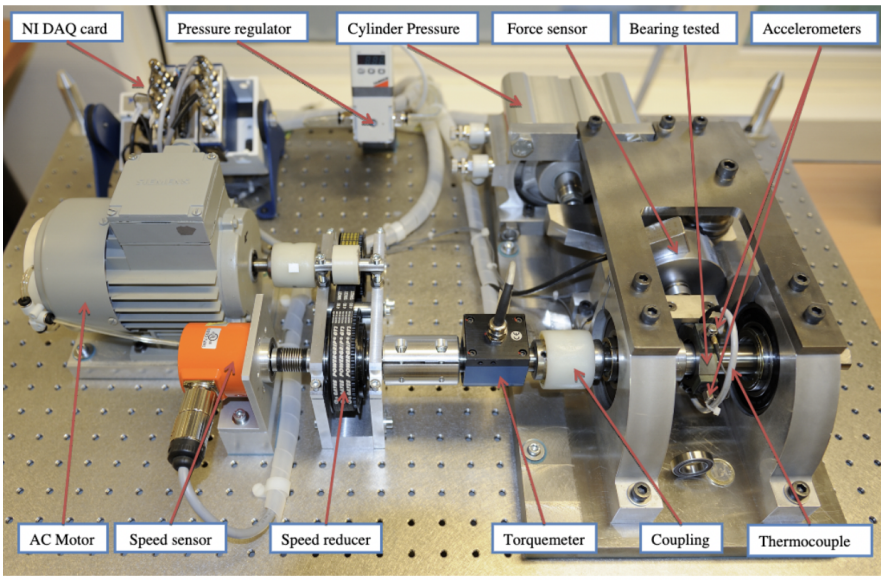


Figure 4.1 – PRONOSTIA data collection and measurement setup [33]

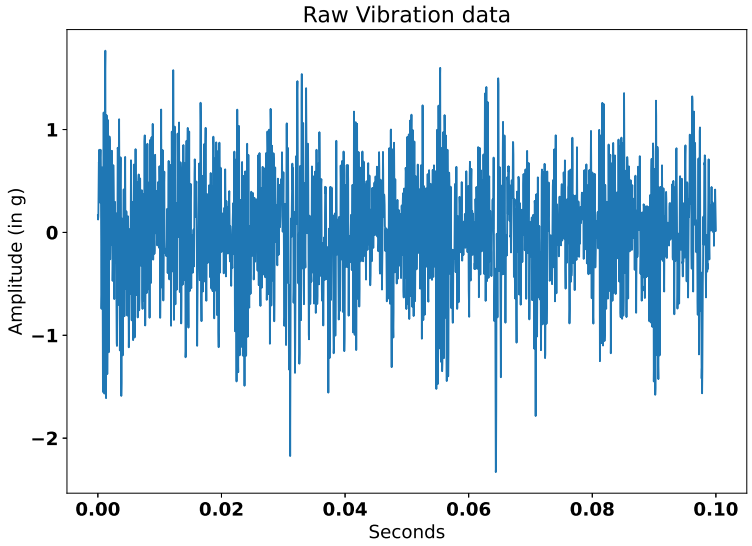


Figure 4.2 – Single sample of second Bearing from Condition 1 (axis I)

The PRONOSTIA dataset gained popularity due to the IEEE PHM 2012 Challenge, which is a challenging problem in the academic and industrial communities to investigate bearing health monitoring for PdM applications using this dataset. This is mainly due to the nature of the collection of data which resembles a real-world scenario from multiple sensors. The accelerometers with two perpendicular axes of vibration collect the raw vibration (time series) data. This vibration data were collected at a sampling frequency of 25.6 kHz with every single scan captured for a duration of 0.1 seconds as displayed in Figure 4.2 , which is

from the first axis of the second bearing measured under the first condition. This 0.1-second duration of the machine is referred to as 'one lifecycle' in the rest of the thesis. These scans are stored and available in CSV (comma-separated values) formats. The metadata from the dataset contains the amplitude information measured in g (force exerted) from the two axis of the accelerometers along with the time at which that particular measurement was taken up to microsecond precision. In Figure 4.3, the raw data measured from the first

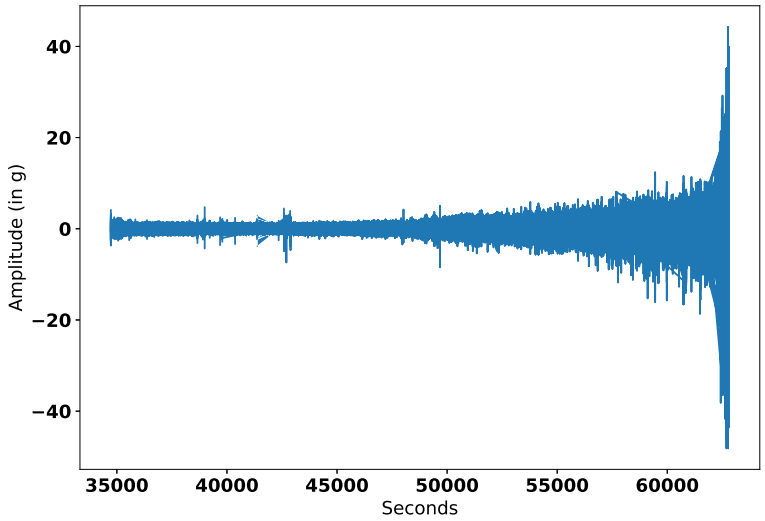


Figure 4.3 – Raw data from healthy state to failure (axis I)

axis is concatenated for the entire duration under the setting of the first condition which starts from the stamp of 35639.14 seconds (resembling a healthy state of the bearing) and ends at the time stamp of 62759.17 seconds (resembling a failure state for the bearing). This entire duration from a healthy state of the machine till failure is described as the Lifetime in the discussion for the rest of the thesis. It can be observed that there is an increasing trend in the data and thereby showing some characteristics in distinguishing the healthy state from failure by observing only the raw data. Furthermore, the dataset was labeled with having each 0.1-second duration of machine life as stored in a CSV file as 1 unit of 'Lifecycle'. Therefore, the total number of such scanned files related to a bearing resembles the complete lifetime of that particular bearing. This information is used for obtaining the labels of the dataset, labeling the first scan with an RUL as the length of the total number of scans and the last file with 1. To run our algorithms, it is important to frame the PdM problem concerning the available data. The PRONOSTIA problem is carried out as a regression problem where the labels are finally normalized to 1 as visualized in Figure 4.4, where 1 resembles a perfectly healthy state and 0 resembles a state of complete failure in the raw data and a fractional value denotes the RUL at that state[52].

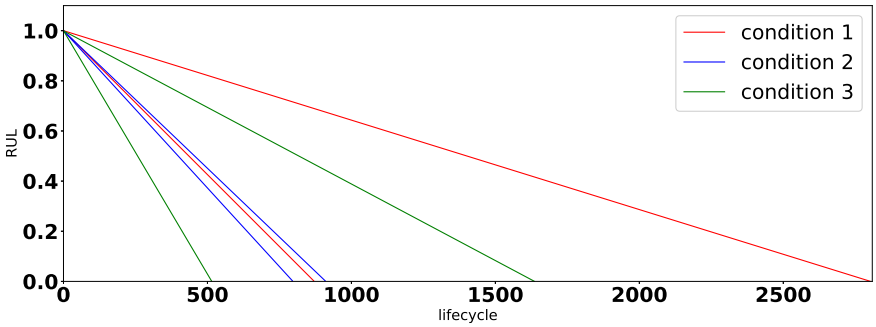
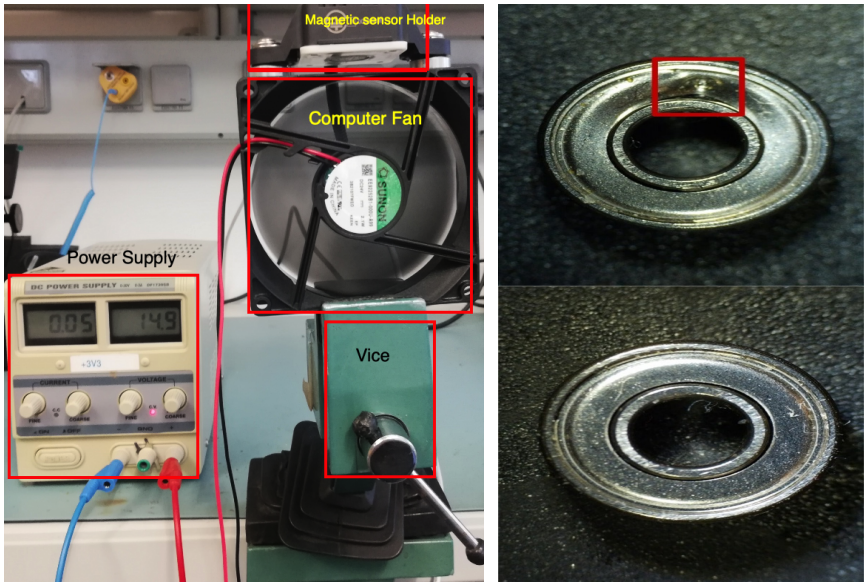


Figure 4.4 – Label visualisation for the training set

Furthermore, for the tasks of FR of this thesis, the final dataset was prepared by using the raw data from all three conditions giving a total single scan of 7534 for the train data and 13959 for the test data. The description of the train and test data is pre-defined according to [33]. However, it might be interesting to note that for any other scenarios involving a prediction of a machine state, the train and test split cannot be conducted randomly. This is because for the training of a ML model in determining machine state, the algorithm needs to learn from every phase of the machine life from its state of health till failure. This dataset is then pushed into the feature extractor to further the investigation of FR.

4.2 FAN Dataset

Systems operating in an industrial setting require a high degree of reliability and availability. Computer fans become useful as it acts as heat sink to cool a particular component. The type of bearing used in a fan can affect its performance and noise. Additionally, fan propellers are used to convey power from a source. Propellers are an array of fan blades that are used for the transmission of power by converting rotational motion into thrust. This is a scenario to investigate bearing and propeller defects for the PdM design of computer fans.



(a) (b)

Figure 4.5 – (a) Setup for the FAN dataset (frontview). (b) Healthy Bearing (bottom) and Defective Bearing (top)

The vibration data has been captured by a three-axis accelerometer at a sampling rate of 2000 samples per second. The sensor resolution of the accelerometer is 1g, where g is the unit for the acceleration. The measurement unit is calibrated in milli g. The dataset was captured at the IMMS with the setup as shown in Figure 4.5(a).

The computer fan is fixed in the vice (also called "schraubstock") shown in green and a magnetic sensor holder as shown in Figure 4.5(a) is used to position the accelerometer to gather the data from its three different axes. The operating voltage levels of the fan are noted from varying from 12 to 23 Volts. The voltage levels of fans are directly proportional to their speed of rotation. Furthermore, this dataset is originally labeled in 3 classes and thereby defining this as a classification problem. There are a total of 9 fans: where one has a bearing defect as displayed in Figure 4.5(b), one with propeller defect, and 7 that are good conditioned. The dataset description has 1580 runs of 0.25 seconds each for the training data and 4741 runs of 0.25 seconds for the test data. This dataset resembles a real-world scenario with more test data. Furthermore, the train data contains 790 runs of good conditioned data, 395 runs each of the two categories of bad conditioned data. For the test data, 3951 runs of good conditioned data are present along with 395 runs each of the two categories of bad conditioned data thereby making the dataset mildly skewed.

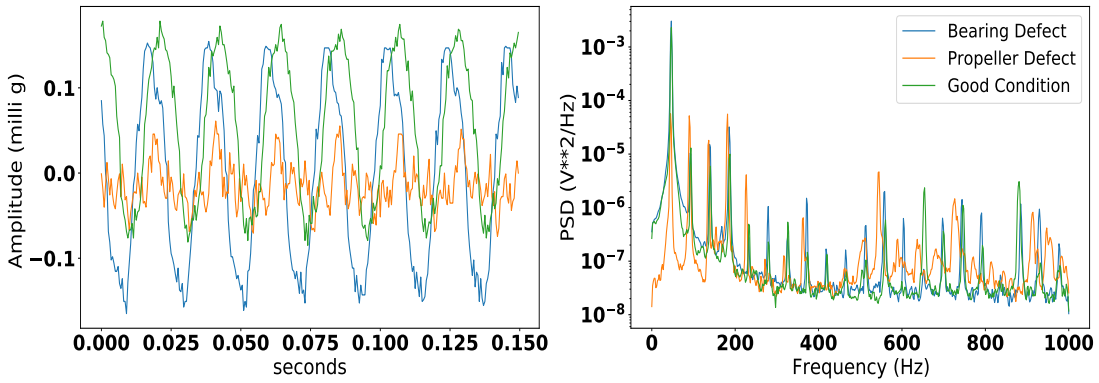


Figure 4.6 – Raw Data Visualisation for Fans in all classes (axis I)

In Figure 4.6, the time-series data captured by the first accelerometer axis for all three classes of fans operating at 20 volts along with the corresponding Power Spectral Density (PSD) to visualize the trends in the frequency domain is depicted. All data has been visualized up to 0.15 seconds and the raw data shows a periodic trend. For the good condition data, a peaky response in the frequency domain representation (PSD) is observed signifying fundamental and decaying harmonics from the corresponding time-domain signal. Also, the highest frequency of operation is 1000 Hz (Nyquist frequency) and multiple peaks can be observed at lower frequencies describing most of the power is within the low-pass of the spectra. For the defective conditions, a trend is observed for lowering amplitudes in the time domain for the propeller defect. Additionally, the peaks dampen in the low pass for both the bearing and propeller defect in the spectra. Therefore, further analysis performed in the frequency domain might reveal a better representation of this data.

4.3 Dataset description

The two above-mentioned scenarios are considered for evaluating the chosen models after FR. The datasets are a good fit for this task at hand as a wide array of features can be extracted using the feature extractors as mentioned in Section 3.1 from both the scenarios in a similar manner. Importantly, the differences in the nature of the datasets are interesting as PRONOSTIA is a regression problem, whereas the FAN dataset is a classification problem. This makes the development of the methodology viable for these two ML problems. Additionally, both datasets describe data that are from a real-world scenario and hence making this investigation a challenging task. The final description of the dataset is described in Table 4.1.

After this raw time-series data is pre-processed using the feature extractors, a wide array of feature sets is observed. A simple test is designed to further them to be utilized for the development of the methodology. The feature extractors are pushed into a simple linear and

Dataset	Description
PRONOSTIA	1. Regression problem with data taken from three different conditions. 2. Dimensionality of train data scans (of length 0.1s each) is 7534 3. Dimensionality of test data scans (of length 0.1s each) is 13959
FAN	1. Classification problem with data is collected in all three classes of the fan 2. Train data dimensionality of scans of 0.25s for each individual classes are: (a) Good condition : 790 (b) Propeller defect : 395 (c) Bearing defect : 395 3. Test data dimensionality of scans of 0.25s for each individual classes are: (a) Good condition : 3951 (b) Propeller defect : 395 (c) Bearing defect : 395

Table 4.1 – Dataset Summary

non-linear model. For PRONOSTIA, all four feature extractors discussed in Section 3.1 are viable for investigation and are therefore proceeded with. However, for the FAN scenario, most of the features extracted from the extractors "except the LFB feature set" performed very poorly and thereby making the rest unviable for further investigation. This makes the LFB feature set useful for further investigation. Finally, the dimensionality of all the four feature sets is summarised in Table 4.2 where broadband features have the lowest dimensionality and DWT features have the highest dimensionality for the PRONOSTIA dataset with a total of 100 features. In the following section, the design of FR algorithms will give insight into establishing a hierarchy of these given features for both of these scenarios.

Dataset	Feature Sets	Dimensionality
PRONOSTIA	1. Broadband Statistics	12
	2. LFB	20
	3. OFB	20
	4. DWT	48
FAN	LFB	15

Table 4.2 – Dimensionality of Feature Sets

Chapter 5

Conceptual Design and Implementation

A theoretical design of the workflow is illustrated in this chapter below followed by a detailed map of the implementation details.

5.1 Conceptual Design

The performance of a deployment device for PdM applications is heavily dependent on the number of input features as that determines the size of the model. Therefore, the design space of an edge computing device is largely characterized by the number of input features. Further, the design space constitutes the hardware cost, which translates to the number of sensors and the number of axes involved in gathering the data. Concisely, there are primarily two areas of improvement from a system design lens: the amount of space complexity involved at the deployment device and the training time involved in executing an algorithm. These two factors are scrutinized during the design of this approach through a reduction of input features. A reduction of input features at the pipeline directly results in improving these performances.

A typical example of a design space from a practical setting is illustrated in Figure 5.1, where we have data from two sensors, each having three axes that capture vibration data. Here, the information collected out of the sensors could be from the broadband or the spectral bands. Therefore, feature sets can be generated through three ways from raw time-series data, by adding a new sensor, by capturing features in the broadband, or a narrowband representation through transforming the data in the frequency or a time-frequency domain.

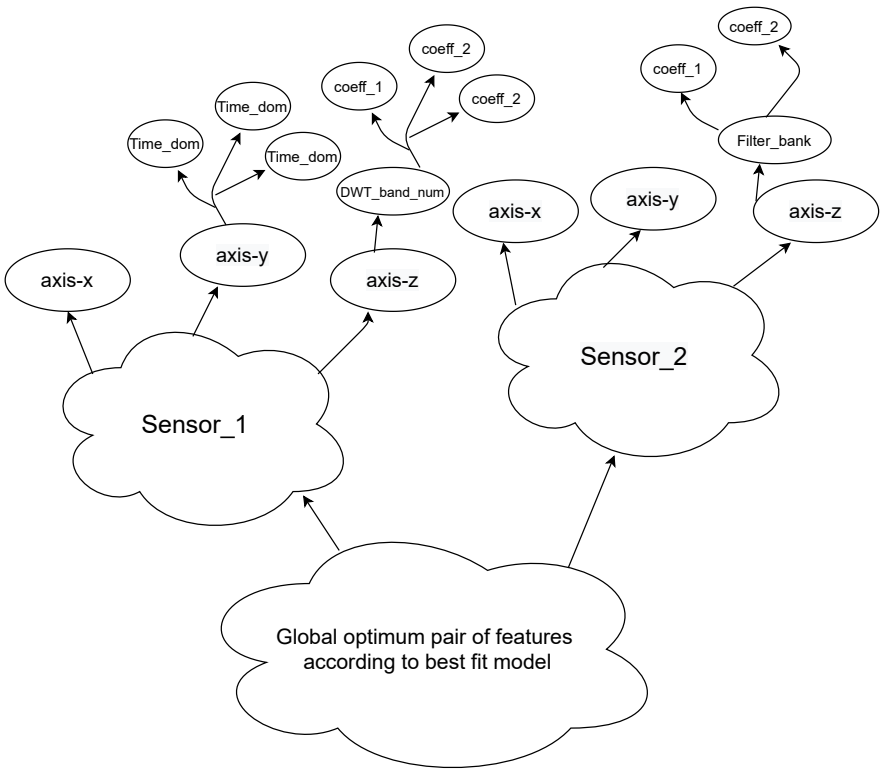


Figure 5.1 – A representation of the design space in PdM applications

As it can be observed from Figure 5.1 that the dimensionality of the possible feature sets can get extremely high which might create a bottleneck at the pre-processing part of the pipeline. In addition to that, it also affects the hardware costs, model size, and computational costs involved in the overall performance. This is the exact problem that the thesis tries to investigate informed by the SOTA in Chapters 2 and 3. Furthermore, a need to establish a comparison methodology is important as it informs a system designer to choose an FR algorithm and the number of features that are optimized for performance for suitable embedded system design in PdM applications. However, this overall methodology with variations in the approach could be expanded further in the development and comparison of a set of FR algorithms for investigating different scenarios involving time-series data.

Design methodology

This sub-section reviews the details of the different configurations of the datasets, feature sets, FR algorithms, and models highlighting the specific details in the planning and design of the experiments to be investigated. The theme of FR and how to conduct performance testing in a given scenario is central to the thesis.

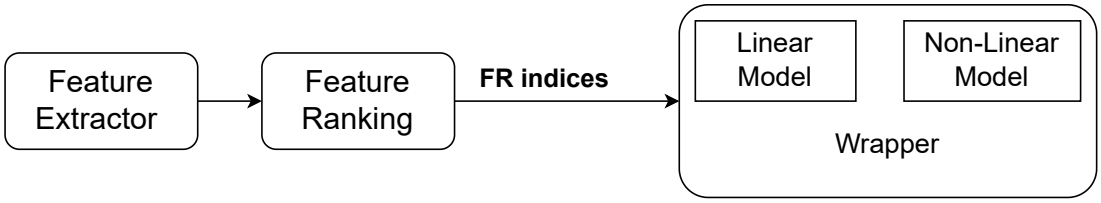


Figure 5.2 – Reduced information processing pipeline for investigation

Firstly, the entire information processing pipeline has been narrowed down to Figure 5.2 containing a feature extraction module, an FR module, and finally a wrapper for comparing the FR algorithms. The wrappers are broadly categorized into linear models and non-linear models as it is only used as a *tool* to compare model performance. Therefore, the choice of wrappers is restricted into a Linear (regression) and Logistic (classification) for linear models and an MLP for a non-linear model. Furthermore, the analysis of the design space is conducted in a bottom-up approach for every feature extractor separately. This means that the system-design decisions for every feature extractor can be based on the performance of each feature set with all possible FR algorithms under examination after the completion of performance testing.

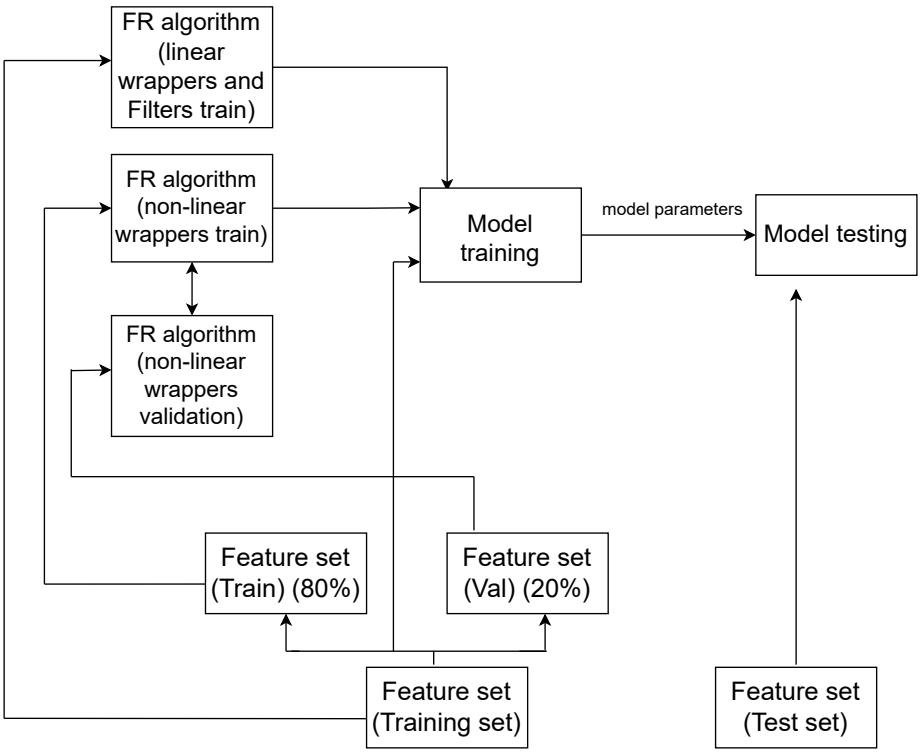


Figure 5.3 – The train and validation splits

Since the methodology is completely data-driven, to design the approach, the strategy of training and testing the individual components is of vital importance. A detailed description

of the exposure of the training and the testing data is illustrated in Figure 5.3. *Overfitting* and *Underfitting* are two of the fundamental challenges that concern the performance of any ML algorithm. Therefore, a cross-validation technique can be employed to understand the behavior of model performance with varying model parameters. Overfitting is more likely with non-linear models that have more flexibility when learning a target function. As such, many non-linear ML algorithms also include parameters or techniques to limit and constrain how much detail the model learns. Therefore, the treatment of training data has to be evaluated for an ML algorithm that needs a careful design of its model hyperparameters. Additionally, for scenarios involving larger availability of training data, multiple techniques can be employed as described in [44]. However, for the fine tuning of hyperparameters of models involved in both the FR algorithms (CO and LASSO) and the MLP, a train-validation split is conducted. This is because LASSO and CO being embedded FR methods require a perfomance evaluation on a validation set for the choice of their optimal hyperparameters. This also prevents the inability of the models to generalize over unseen test data. Additionally, for the ML model to learn every phase of the life-cycle of a machine or every class of the machine state, the train-validation splits need to be performed in a certain manner[44].

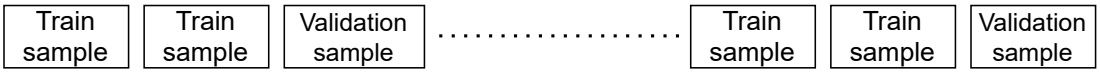


Figure 5.4 – The train and validation splits

This is illustrated in Figure 5.4, where the validation data is downsampled from the entire sequence of the overall training data. This ensures that the ML model is exposed to learning from various segments of the life cycle of a machine and ensures a better chance of generalizing over unseen data for validation. Further, the chosen metric for evaluating the validation data is Root Mean-Squared Error (RMSE) with a Mean-Squared-Error (MSE) loss function for the PRONOSTIA scenario. For the final evaluation for the comparison of all FR algorithms, RMSE is chosen as the metric as it relates to the standard deviation of the error in RUL prediction. For the FAN scenario, due to the nature of the slightly imbalanced dataset, the accuracy and F1 score are chosen (with macro averaging) as metrics over the validation data. This is to make the model sensitive to False Positives (FP) and False Negatives (FN)[6, 51].

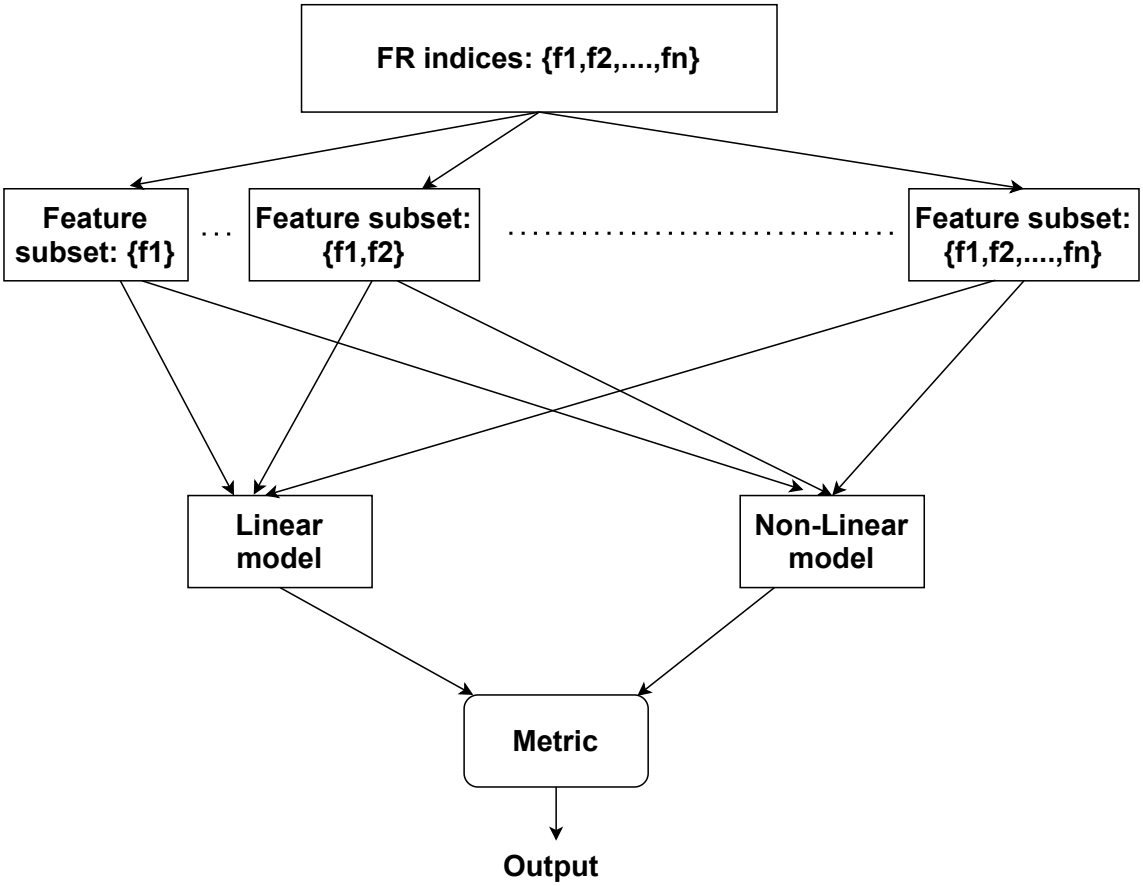


Figure 5.5 – Design for comparison of various FR algorithms

Finally, another important component of the design is described in Figure 5.5 which highlights the mechanism of comparing FR algorithms using a wrapper. Every FR algorithm offers an array of FR indices at the output denoted as {f1,f2,...., fn} sorted from highest to lowest importance, where f denotes the feature index. All FR algorithms except RFE give one order of feature importance and are ranked in descending order from best to worse. RFE however offers explicit FR and hence every entry of the FR index at its output distinctly mentions the features involved for that particular rank. Further, to observe the feature dimensionality impact on the model's performance, features from each rank need to be pushed into a wrapper. This process is used to train on the training data recursively with sequentially choosing the subset of features by rank as shown in the diagram. The output of the wrapper is obtained from the test data with the same metric for evaluation. Consequently, the fairness of this method is obtained in three ways:

1. The test data is first pre-processed and the corresponding feature sets are given *identically* in the order of FR to both the models with no modifications.

2. Each individual model (linear and non-linear) during the test has the *same set of design hyperparameters*.

3. The metric for evaluating the test set is same for all feature sets and models under a defined ML scenario.

Lastly, after obtaining the result of a performance score for each feature rank, from a system design point of view, the majority of the features are noted after which the curse of dimensionality trend is observed. An analysis is performed on how these features would improve the system design through FR.

Due to the stochastic nature of neural networks, to monitor the performance, Monte Carlo simulations are conducted by obtaining multiple realizations to find out the statistical properties of the obtained performance[59]. Despite the conceptual and algorithmic simplicity, the computational cost associated with a Monte Carlo simulation can be staggeringly high. Due to this a total of 10 realizations have been conducted for monitoring the performance of every non-linear model. Further, this is also conducted while computing the timing performance involved in every FR algorithm.

The 95% confidence intervals are calculated according to the t-distribution statistics as described in Equation (5.1) for the visualization of the performances of the MLPs[60]. Here $\bar{X}$ is the random variable of the measured performance, μ is the mean of the realizations, s is the standard deviation of the measured performance and n is the degrees of freedom which is one less than the number of realizations. The t-score is calculated with the help of the t-distribution table for 9 degrees of freedom and a confidence score of 95%.

$$t = \frac{\bar{X} - \mu}{s/\sqrt{n}} \tag{5.1}$$

5.2 Implementation

For the development of the methodology, the entire implementation has been done using Python 3.7.12 in conjunction with the help of existing ML and DL frameworks like *sklearn, pandas, TensorFlow, keras* and the APIs involved in performing FS. The code snippets for some of the implementation are mentioned in Appendix C

5.2.1 Feature Extractor implementation

Algorithm 3 Feature extraction algorithm

Input data_x
Output broadband_x, dwt_x, lfb_x, octavefb_x

Require: data_x ▷ raw time series full Lifetime data from axis I (in frames)
Ensure:
broadband_x ▷ stores the broadband statistics
{lfb_x, octavefb_x, dwt_x } ▷ stores narrowbandfeatures
{dwt_num, OFB_order, num_LFB_features, num_OFB_features} ▷ narrowband feature specifications
FFT_bins ▷ frequency domain resolution in normal scale
extract_all(data) ▷ calculates statistical features out of raw data
wavelet(data, number of bands) ▷ calculates DWT band transformed data
LFB(data, num_LFB_features, FFT_bins, sampling_rate) ▷ calculates total LFB features from raw data
OFB(data, num_OFB_features, OFB_order) ▷ calculates OFB features from raw data
procedure CALCULATE BROADBAND AND NARROWBAND FEATURES
    for frames ∈ data_x **do**
        broadband_x ← extract_all(frames) ▷ stores all broadband features
        lfb_x ← LFB(frames, num_LFB_features, FFT_bins, sampling_rate) ▷ stores all LFB features
        octavefb_x ← OFB(frames, num_OFB_features, OFB_order) ▷ stores all OFB features
        wavelets ← wavelet(frames, number of bands) ▷ stores samples for all DWT band
        for index_dwt ∈ dwt_num **do**
            dwt_x ← extract_all(wavelets[index_dwt]) ▷ stores all DWT statistical features
        end for
    end for
end procedure

Figure 5.6 – Broadband and narrowband Feature extaction algorithm

The description of the feature extraction procedure is summarized in Figure 5.6. To generate the broadband statistics from the raw data some numpy functions are utilized. The window length (frames) of the signal for both scenarios varies according to the description of the datasets. There are 6 statistical broadband features obtained as mentioned in Section 3.1.1 to carry out the broadband analysis and thereby obtain a dimensionality of 12 features for the PRONOSTIA scenario. This process is described by the 'extract_all function' in Figure 5.6.

Followingly, the DWT features are obtained using the PyWavelets module[1]. For this thesis, the statistical features extracted out of the specific bands that are calculated by the DWT are used as features for this feature set design. For the PRONOSTIA dataset, the DWT has 4 levels of decomposition with 6 statistical features (as mentioned in Section 3.1.1) extracted out of each specific band for two accelerometer axis thereby giving a total dimensionality of 48 features. These levels are also referred to as spectral bands in the rest of the thesis.

[1] `https://pywavelets.readthedocs.io/en/latest/`

Further, the LFB features are obtained by utilizing the python speech features module[2]. The function calculates the filter bank power[34]. In other words, first, the signal is divided into shorter frames and for each frame, the spectogram is calculated. Further, the mel filterbank is applied to the power spectra and the logarithm of all filter bank energy is taken as a feature. This can be easily achieved using the python speech features package.

The OFB (which are Butterworth IIR filters) are made using a combination of numpy and scipy functions. For the PRONOSTIA scenario, a sample rate of 25.6 kHz for a window length of 0.1 seconds and the number of filters used for the entire spectrum is 10 with 4096 Fast Fourier Transform (FFT) points for obtaining the LFB feature set. The filters are applied to both the accelerometer axis and the filter bank power is used as a feature. This gives a total dimensionality of 20 features for the LFB feature set. Lastly, the total dimensionality of the OFB feature set is 20 features, where 10 coefficients come from each axis. Each coefficient describes the filter bank power for that specific spectral band.

For the FAN dataset, the LFB feature set is obtained by applying 5 filters on the entire spectrum on all three axes with a 2KHz sampling frequency and 512 FFT points. Each filter bank power is used as a feature. This gives a total dimensionality of 15 features.

5.2.2 Feature ranking algorithm design

This subsection details the parameters and hyperparameter design flow of the FR algorithms as mentioned in Chapter 2.

PCA

In practice, for PCA usually only the top 4-5 eigenvectors are especially descriptive, accounting for no more than 60-85% of the variation, depending on how noisy the dataset is. The remaining eigenvectors add little value (or information in terms of describing the data). That's why it is attempted to describe most but not all of the variation by combining only the primary components, which account for as much of the variation as is meaningful to the scenario, which certainly will be less than 95%. Therefore, 95% of the explained variance ratio is considered for both feature sets and the eigen vectors are scaled accordingly and summed to obtain a resultant vector whose weights corresponds proportionally to feature importance.

The PCA components (explained variance ratios and the corresponding eigen vectors) are calculated with the help of the sklearn module. The details of the FR algorithm is provided in Figure 5.7. The *absolute values of the indices of the entries correspond to feature importance*. Additionally, the resultant vector obtained involving 95% of the variation in data is also a feature ranking metric for this filter method as it corresponds to its distribution

[2] `https://python-speech-features.readthedocs.io/en/latest/`

Algorithm 4 PCA FR Implementation

```
    Input X_train
    Output FR_indices
Require: X_train                                          ▷ Training feature set
Ensure: variance_ratio_list ← List of explained_variance_ratio after EVD
 1: PCA_weights ← List of Principal Components (PC) after EVD
 2: FR_weight ← Resultant FR_weights after PCA
 3: FR_indices ← Ranked FR indices for PCA
 4: procedure CALCULATE PCA_WEIGHTS
 5:     pca ← PCA(0.95)                          ▷ operator computes PC upto 95%
    explained_variance_ratio
 6:     variance_ratio_list,PCA_weights←pca.fit_transform(X_train)
 7:     num_components ← len(variance_ratio_list)
 8:     for index ∈ num_components do
 9:         FR_weight + = variance_ratio_list[index]*(abs(PCA_weights[index]))
10:     end for
11:     FR_indices = argsort(FR_weight)          ▷ The sorted FR indices of the
    FR_weights after taking absolute values
12: end procedure
```

Figure 5.7 – PCA FR algorithm

statistics. However, this algorithm lacks exclusivity in FS as it offers an overall ranking without any conduction of training.

RFE

Algorithm 5 RFE FR Implementation

```
    Features X_train
    Labels y_train
    Output FR_indices
Require: X_train, y_train                          ▷ Training feature set and label
Ensure:
 1: linear_model ← The linear model to wrap around RFE
 2: RFE ← operator that selects i features with the help of linear_model
 3: FS_i ← 'i' best selected features by RFE
 4: FR_indices ← ranked features for every index by RFE
 5: procedure CALCULATE RFE FR_INDICES
 6:     num_feat ← total number of features in the feature set
 7:     for index ∈ num_feat do
 8:         lr = linear_model                   ▷ choose model to wrap around RFE
 9:         rfe = RFE(estimator= lr, step=1, n_features_to_select=index)
10:         rfe.fit(X_train, y_train)
11:         rfe.get_support()                   ▷ returns True values for selected features
12:         for i,j ∈ enumerate(rfe.get_support()) do
13:             if j then:
14:                 FR_i.append(i) ▷ stores the indices for choosing best features
    per rank
15:             end if
16:         end for
17:         FR_indices.append(FR_i)
18:     end for
19: end procedure
```

Figure 5.8 – RFE FR algorithm

A linear model is fed as a wrapper for both the scenarios in RFE to simplify the complexity of this algorithm. A Linear Regression wrapper is chosen for regression problems and a Logistic Regression wrapper is chosen for classification problems. The weights of the linear model is considered as the *feature importance* for the evaluation in RFE. The details of the algorithm is given in Figure 5.8. The sklearn module helps to find the optimal set of features for a specified number of features and a model to wrap RFE around with and thereby conducts FS only. In order to perform FR this process is done iteratively in a loop with varying the number of features from 1 to the maximum number of features in a given feature set and thereby resulting in a sequence of feature indices contributing to performance for every rank.

LASSO

The equivalent design of LASSO for a classification problem would be similar to a Logistic regression model with regularisation. However, for a multiclass problem this yields to metrics that favour FS by offering weights to the feature indices for each particular class separately. As a consequence, it could be useful for FS but cannot be utilized for FR in solving classification problems. Therefore, the investigation of LASSO is excluded for studying the FAN dataset.

The regularisation parameter λ is explored for optimal tuning in the range of 0.01 to 1 with 50 evenly spaced samples in between this range as described in Figure 5.9. Further, a value is chosen for λ after which there are no accelerating returns in performance and training time. This is because since LASSO is an embedded method, it conducts FR and model output simultaneously. Therefore, the hyperparameter design for λ is evaluated by the overall model performance. Finally, the LASSO coefficients are computed with the help of sklearn module after training for a particular feature set. The *absolute values of the coefficients are the metric for LASSO's feature importance*. This is because both negative and positive coefficients indicate feature importance. These coefficients are sorted after taking the magnitude of these values and arranged in descending order and thereby achieving FR.

CancelOut

Since DL is a very computationally expensive phenomenon as it requires multiple training of models with large potential design spaces for parametric and hyper-parametric tuning, the design of the hyper-parameters and parameters becomes challenging.

Figure 5.11 – CO FR model description (PRONOSTIA)

Algorithm 6 LASSO FR Implementation

Features X_train
Labels y
Output FR_indices

Require: X_train, y ▷ Training set feature and label pair
Ensure: (x_train,y_train), (x_val, y_val) ▷ Training and validation sets feature and label pair

alpha_range ← 50 evenly spaced points between 0.01 and 1
time_LASSO ← stores the training time for every value of hyperparameter scan
RMSE_LASSO ← stores the RMSE on the validation set for every value of hyperparameter scan
linear_model.lasso ← LASSO model that takes regularization value alpha with a fixed number of maximum iterations
linear_model.lasso.coef_ ← feature importance metric by LASSO
FR_indices ← ranked features for LASSO
alpha_optimal ← optimal regularization coefficient
procedure CALCULATE OPTIMAL REGULARIZATION PARAMETER AND LASSO FR_INDICES
    for alpha_val ∈ alpha_range **do**
        start = time() ▷ starting time
        lasso = linear_model.Lasso(alpha=alpha_val,max_iter=20000)
        lasso.fit(x_train,y_train)
        selected_feat_indices = argsort(abs(lasso.coef_)) ▷ returns boolean values for selected features
        end = time() ▷ end time
        y_pred = lasso.predict(x_val) ▷ predicted value over validation data
        RMSE_LASSO.append(sqrt(mean_squared_error(y_pred,y_val)) ▷ performance on validation data
        time_LASSO.append((end-start))
    end for
    choose alpha_optimal from alpha_range to jointly optimize FR training time and performance
    lasso = linear_model.Lasso(alpha=alpha_optimal,max_iter=20000)
    lasso.fit(X_train,y)
    FR_indices = argsort(abs(lasso.coef_)) ▷ The LASSO FR indices
end procedure

Figure 5.9 – LASSO FR algorithm

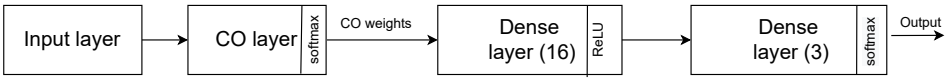


Figure 5.12 – CO FR model description (FAN)

For this reason, a simplistic design approach is directed for both scenarios. Firstly, in conjunction with the CO layer (as described in Section 2.3.2) at the front-end, an MLP is considered as a model for wrapping. A number of hidden layers in the overall CancelOut model is described for PRONOSTIA in Figure 5.11 and in Figure 5.12 for the FAN scenario with the optimal hyperparameters explored in Table 5.1. Two important parameters for FS

Algorithm 7 CO FR Implementation

Features X_train
Labels y
Output FR_indices

Require: X_train, y ▷ Training feature set and label
Ensure: (x_train,y_train), (x_val, y_val) ▷ Training and validation feature sets and labels

lambda_1_range, lambda_2_range ← 10 evenly spaced regularization parameters between 0 and 1
time_CO ← stores the training time for every value of hyperparameter scan
perf_CO ← stores the value of the performance metric for a given ML problem
procedure CALCULATE OPTIMAL REGULARIZATION PARAMETER AND CO FR_INDICES
    for l1 ∈ lambda_range_1 **do**
        for l2 ∈ lambda_range_2 **do**
            start = time() ▷ starting time
            Initialize CO layer ←activation='sigmoid', l1, and l2
            number of hidden layers/final layer on type of ML problem ← optimal set of hidden layers after grid search separately
            Compile model with (x_train,y_train) and perform validation over (x_val, y_val) with a chosen metric on the type of ML problem
            FR_indices ← argsort(model weights of CO_layer) ▷ sort model weights of CO
            end = time() ▷ end time
            time_CO.append((end-start))
            perf_CO.append(performance scores over validation data)
        end for
    end for
    choose lambda_1_optimal and lambda_2_optimal from the hyperparameter range to jointly optimize FR training time and performance
    Use lambda_1_optimal and lambda_2_optimal values in the CO for performing FR
end procedure

Figure 5.10 – CO FR algorithm

(λ_1 and λ_2) are explored within a range of 0 and 1. This is obtained through a grid search by having 10 equally spaced points between 0 and 1 for both these parameters as described in Figure 5.10. Further, the nature of the performance is observed with the variation in these two parameters with more resolution to observe changes in the trends of the performance, and the best and worst performances are noted for both scenarios. The *CancelOut weights are the metric for feature importance for this method*. These weights are then arranged in descending order of importance.

Dataset	FR Algorithm	Hyperparameter description
PRONOSTIA	1. PCA 2. RFE 3. LASSO 4. CancelOut	Explained variance ratio upto 95% Linear regression wrapper $\lambda \in [0.01, 1]$ • num_layers:$\{2, 3\}$ • num_neurons: $\{16, 32, 64, 128\}$ • $\lambda_1, \lambda_2 \in [0, 1]$
FAN	1. PCA 2. RFE 3. CancelOut	Explained variance ratio upto 95% Logistic Regression wrapper • num_layers:$\{2, 3\}$ • num_neurons: $\{16, 32, 64, 128\}$ • $\lambda_1, \lambda_2 \in [0, 1]$

Table 5.1 – Hyperparameter design summary of FR Algorithms

A summary of all the design parameters of the FR algorithms is presented in Table 5.1 which is further investigated in Chapter 6 with these specifications and optimal parameters are used in the development of the comparison methodology. Additionally, a shuffle is also conducted of the feature sets and the shuffled indices are identically treated for evaluating the performance of every wrapper for each feature set at the end of the pipeline. This is done in order to analyze how a random chance affects the model performance without performing any FR. Due to limitation in time, the stochastic gates method has not been implemented[35]. However, it is an interesting choice for conducting FR.

5.2.3 Models Design specifications

For the PRONOSTIA scenario, the linear model is chosen as a Linear Regressor and the non-linear model chosen is an MLP. Likewise, for the FAN scenario, the linear model chosen is a Logistic regression classifier ("one vs rest") and the non-linear model chosen is an MLP.

Dataset	Model	$1/\lambda$	regularizer	optimizer	number of layers	number of neurons
PRONOSTIA	Linear regression	$(10^{-5}, 10^{2})$	$l2$	{'svd','cholesky','lsqr','sag'}	N/A	N/A
	MLP	N/A	dropout rate 0.5	{'Adam','RMSprop'}	{3, 4}	{16, 32, 64, 128}
FAN	Logistic regression	$(10^{-3}, 10^{3})$	$\{l1, l2\}$	{'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'}	N/A	N/A
	MLP	N/A	dropout rate :0.5	{'Adam','RMSprop'}	{3, 4}	{16, 32, 64, 128}

Table 5.2 – Hyperparameters of the final wrapper

Hyperparameter tuning

For the exploration of the optimal hyperparameters in an ML model there exists a wide array of popular methodologies like Bayesian search, Grid search and Random search[48, 49]. This is primarily done to test the effectiveness of the model over unseen test data by using a cross-validation procedure. Out of these possibilities, a simple grid search methodology is chosen for this thesis as it is easily realizable and provides insights into how the hyperparameters affect each other. Additionally, grid search is conducted with a structured and controlled optimization strategy. However, the fundamental drawback of a grid search method is that it is computationally very expensive. Therefore, in the vast design space of hyperparameters for the given PdM problem, only a certain set of parameters are utilized for the grid search optimization as illustrated in Table 5.2.

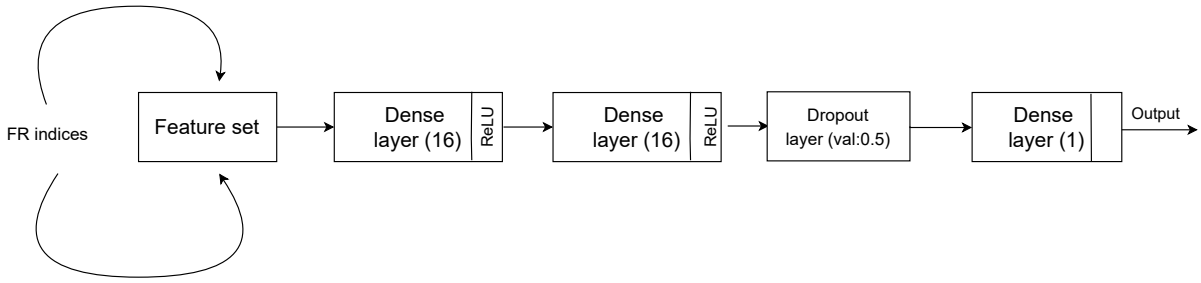


Figure 5.13 – MLP description for PRONOSTIA

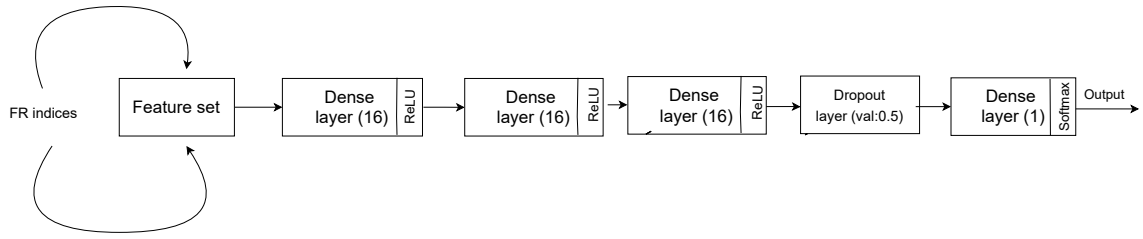


Figure 5.14 – MLP description for FAN

For the linear models, the optimizer, regularisation coefficients, and regularisation penalty are explored. This is done using the 'GridsearchCV' module in sklearn with a 10-fold cross validation procedure. This is done only to choose the range of hyperparameters that helps solve the ML problem at hand (regression or classification) and can affect a linear model's performance. However, multiple effective strategies with variations in the train and validation data can be investigated as future work.

For the MLP since the scope of exploration of hyperparameters is extremely high, a strategic methodology is devised before performing a grid search. Also, the optimizer chosen is fixed to 'RMSprop' with a learning rate of 10^{-2} for the FAN scenario and 10^{-3} for the PRONOSTIA scenario after one time testing over the LFB feature set. The description of an instance of MLP models for both the scenarios is displayed in Figures 5.13 and 5.14. In both the figures, the models have an input of feature subsets followed by a set of dense layers with a certain number of neurons per layer and finally followed by a dropout layer and an output layer. For the PRONOSTIA scenario, the metric of evaluating performance investigated is RMSE and MAE during training as it informs the system directly how erroneous is the output from the ground truth. However, for the final evaluation and comparison methodology, RMSE is chosen for the discussion. For the FAN dataset, a similar approach is conducted with the change of the evaluation metric as F1-score with macro-averaging to reflect the imbalance of class distributions. The metrics are further described in Appendix B

Since a neural network is an universal function approximator, it can fit the data into a function at risk of overfitting. Therefore, the scan of the number of parameters of the NN is of crucial importance for the performance of the model. Also, deeper neural networks often compute a better representation of the data as shallow networks with more number of neurons[56]. Hence, the number of layers and the number of neurons in each layer is chosen as a hyperparameter in the grid search as mentioned in Table 5.2. The results of the MLP wrapper on the optimal FR indices are used for evaluating performance on the validation data for choosing the final hyperparameter setting.

The MLP is trained with a batch size of 512 for 200 epochs with an Early stopping of monitoring the validation loss with a patience parameter of 50 and a minimum change of 0.05 is monitored as a threshold to qualify as an improvement for both scenarios. The

training, validation, and test scores are stored after training the MLP for every feature rank (for each feature set) for comparison at the end.

Since the FAN dataset is slightly imbalanced with a 2:1:1 ratio in the training data, a wide array of over-sampling and under-sampling techniques could be used as described in [45, 46, 47] out of which SMOTE (Synthetic Minority Oversampling Technique) is used. The mechanism is displayed in Figure 5.15

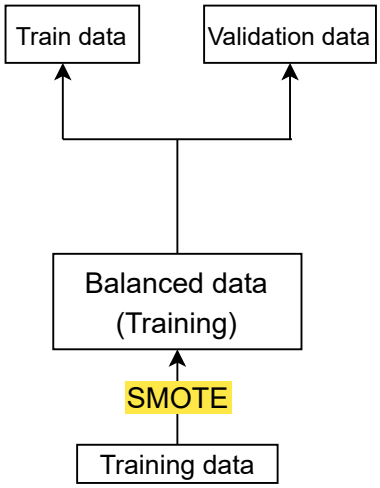


Figure 5.15 – Technique to approach imbalanced datasets

This is because there is a loss of data using undersampling and oversampling using SMOTE balances the dataset using synthetic data. Further, in real-world scenarios like the one investigated the distribution of healthy machine classes to unhealthy machine classes are generally unbalanced. Hence, this method tries to mitigate this problem with additional synthetic data. However, since the dataset is not heavily skewed, accuracy could also be investigated as a performance metric and the investigation of the curse of dimensionality could also be conducted by this metric. Also, accuracy and F1 score (micro averaged) for multi-class classification are the same and accuracy as a metric is easily interpretable.

Algorithm 8 Final wrapper Implementation

Features X_train, X_test
Labels y, y_test
Input FR_indices, (X_train,y), (X_test, y_test)
Output performance scores on test data

Require: (X_train,y), (X_test, y_test) ▷ Training and test feature label pair
(x_train,y_train), (x_val, y_val) ▷ Training and validation feature sets and labels
FR_indices ▷ FR indices in descending order of importance
Ensure: model ▷ Linear or non-linear wrapper on which data needs to be tested
{h1,h2,h3...hn} ← hyperparameters to be scanned in a grid search
h_optimal ▷ optimal hyperparameters to conduct model training over test data
num_feats ← The total number of features in a feature set
procedure CALCULATE OPTIMAL HYPERPARAMTERS AND PERFORM TESTING WITH FR_INDICES
    Perform grid search on training model with (x_train,y_train) and validating on (x_val, y_val) ▷ gives h_optimal from {h1,h2,h3...hn}
    model ← assign design parameters with h_optimal
    feat ← null ▷ appends features according to ranking to obtain performance per ranking
    y_pred ▷ predicted value of ML model on test data
    perform(y_test,y_pred) ▷ returns a performance value on test data according to specific metrics
    for index ∈ length(num_feats) **do**
        feat.append(FR_indices[index])
        model.fit(X_train[:,feat],y) ▷ train model with
        features per rank on training data
        y_pred = model.predict(X_test[:,feat)
        perform(y_test,y_pred)
    end for
end procedure

Figure 5.16 – Final wrapper for testing

Finally, after the selection of the model, a wrapper is implemented where the FR indices are looped into the model for model training as described in Figure 5.16 and the performance over the test data after preprocessing is conducted for every feature index according to FR. This performance score is used for evaluating the FR algorithms in Chapter 6.

Chapter 6

Results and Discussion

This chapter illustrates the findings obtained while implementing the FR algorithms and the testing phase of the developed methodology followed by a discussion on some of the key aspects of the results .

6.1 Results

An analysis is conducted on these findings first for the PRONOSTIA scenario followed by the FAN scenario. The analysis involves the comparison of how each of these FR algorithms performs under one common metric and how that might be beneficial to a deployment system performance through a reduction of features.

6.1.1 PRONOSTIA

All feature sets contain only features from vibration data as described in Section 3.1 as any other feature as the current lifecycle of the bearing is excluded as it offers a bias to the problem of FR at hand which is directed to optimize current resources for improving system deployment performance. Additionally, current lifecycle information of the bearing is oftentimes hard to document into a dataset. First, the feature sets are visualized followed by the discussion on the methodology conducted.

Feature Extractor

Broadband Statistics

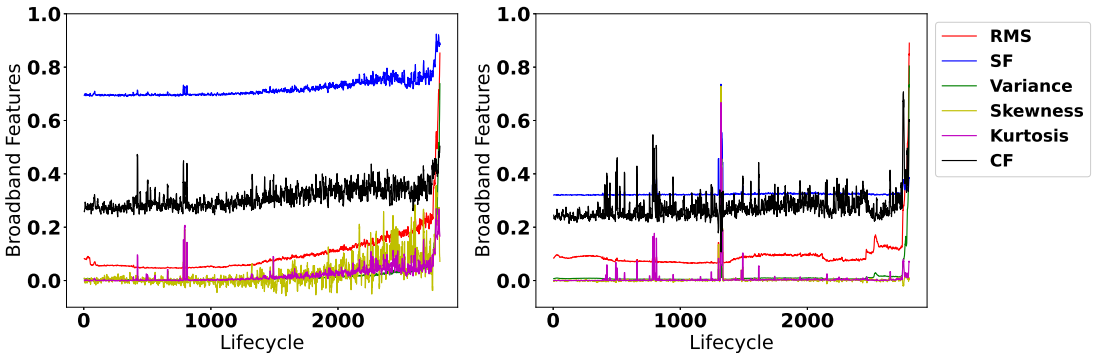


Figure 6.1 – Broadband statistics (left - axis I and right - axis II)

In Figure 6.1, the broadband statistics are visualized for the first bearing under the first condition for both axes from a healthy state to failure. The plot is normalized and then modified with a moving average of 5 samples for better visualization. An increasing trend could be observed for all the features in the first axis, with the least variance on the SF while the RMS had an abrupt increasing trend at the time close to failure. For the second axis information, the trend in the statistical coefficients is noisy with intervallic peaks for the CF and the SF. The second axes Kurtosis has peaks all throughout the machine life with minimum amount of noise. However, RMS has a steady mean with an abrupt increment at the time close to failure.

Sprectral features

DWT statistics

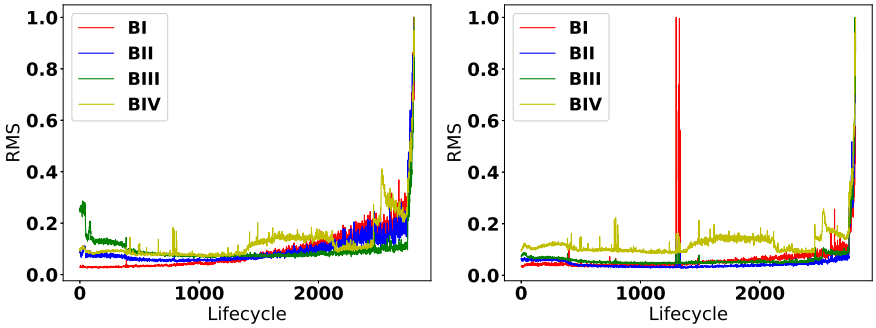


Figure 6.2 – RMS value for 4 DWT bands (left - axis I and right - axis II)

The RMS coefficient is visualized from the training set of PRONOSTIA for the first bearing under the first condition involving both accelerometer axes in Figure 6.2 for all the 4 levels of the DWT. The plot is normalized (linear scale) and an increasing trend is observed signifying higher levels of energy suggested by the RMS at a point of failure. Also, for the second axis, the first band (low pass information (BI)) portrays peaks at the middle of the lifetime of this bearing. This reflects that trends in RMS for both axes might be an useful predictor of the machine state.

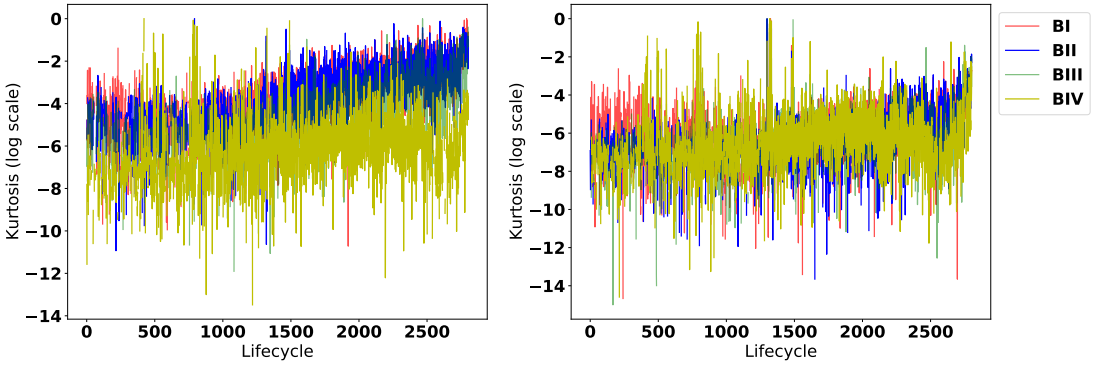


Figure 6.3 – Kurtosis value for 4 DWT bands (left - axis I and right - axis II)

Similarly, the Kurtosis information for the same bearing is visualized in logarithmic scale in Figure 6.3 and the low pass information (BI) for the first axis displays an increasing trend with a 2dB increment from a healthy state to failure. However, mild changes are noticed in the trends of all the four bands in the second axis information.

LFB features

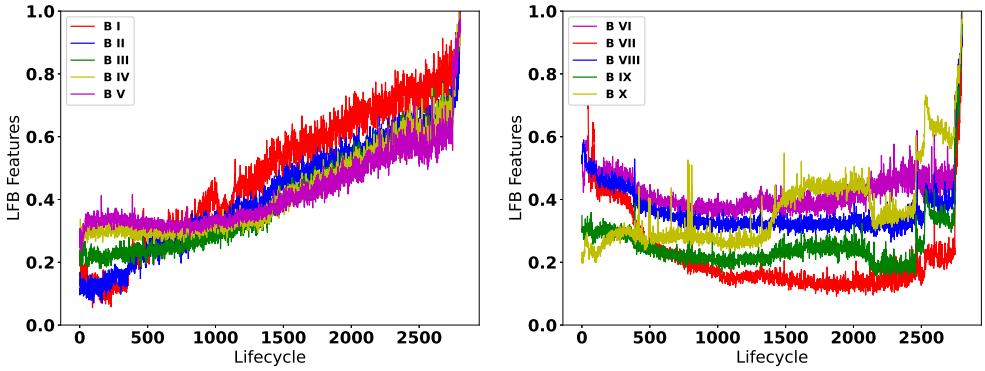


Figure 6.4 – LFB domain PRONOSTIA

The first axis LFB features (normalized) are visualized in Figure 6.4, where all the first five bands representing lower frequencies show an increasing trend with a large variance for the first band. The last five bands show a decreasing trend initially with a intermediate

peaks and during the final few cycles of the bearing it shows a rapid increase. For the second axis LFB features as displayed in Figure 6.5, a similar trend is observed with peaks in the neighborhood of the 1500 lifecycle for the first five bands, and more variation is observed in the highest frequency band (Band X) in the neighborhood of the 1500 lifecycles for the same bearing. The overall relationship of this feature to the machine state is quite prominent for both the sensor axes. There is a linear trend with the RUL and can be an important predictor for machine state information.

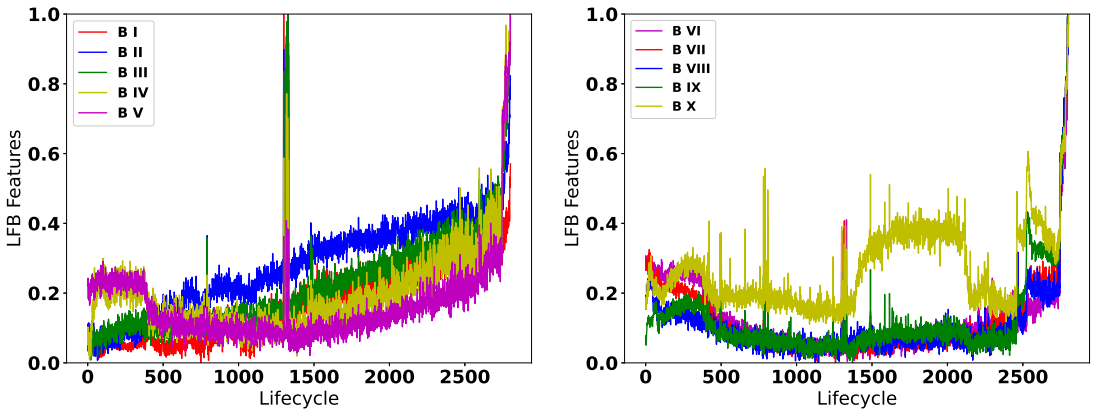


Figure 6.5 – LFB domain PRONOSTIA

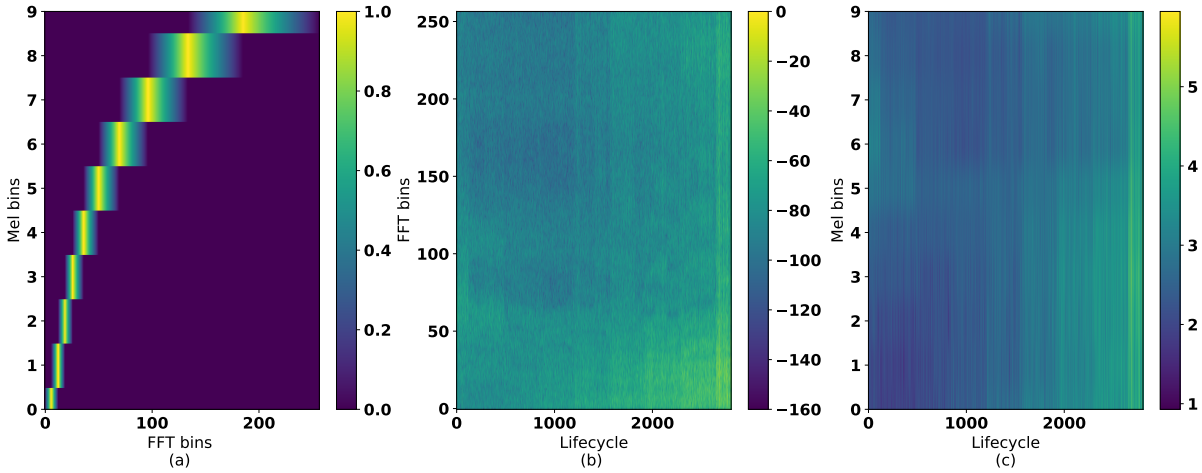


Figure 6.6 – LFB features and Spectogram

In Figure 6.6(a), the triangular filter banks are visualized with the 256 FFT bins and the matrix is sparse with the entries of the filter bank weights present at Mel bins with varying FFT bin length as explained in Section 3.1.3. The spectrogram for the first bearing from the training set for its entire lifetime is visualized in Figure 6.6(b) with power in each FFT bins and the dimensionality of this feature space is very high due to high number of FFT bins (256). Further in Figure 6.6(c), the resultant features obtained have low dimensionality of

10 features and the features obtained are less noisy. Also, high FB power can be observed for all Mel bins.

OFB features

In Figure 6.7, the first axis of this feature set (normalized) is visualized for the same bearing and a high variance in the feature set is observed in the first five bands of the OFB in the lower spectrum. While an increasing trend with a sharp increase is observed in the last five bands of the higher spectrum with an additional peak around the 1000th lifecycle.

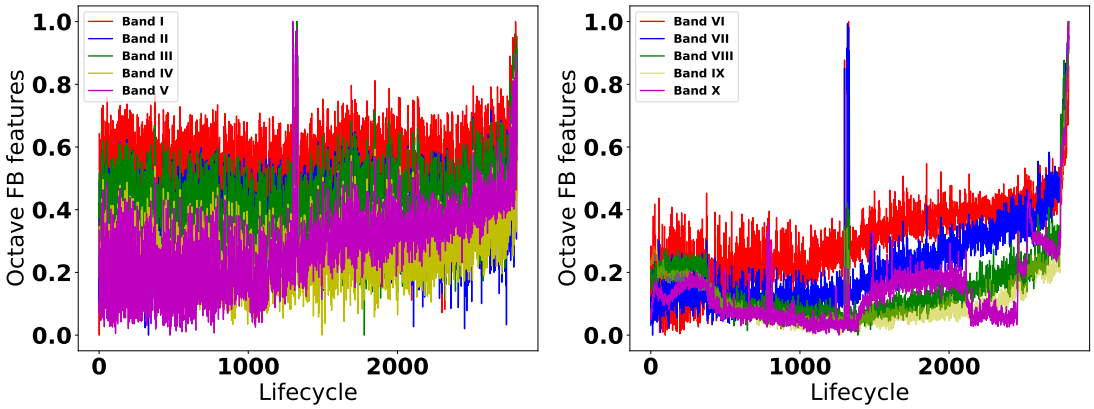
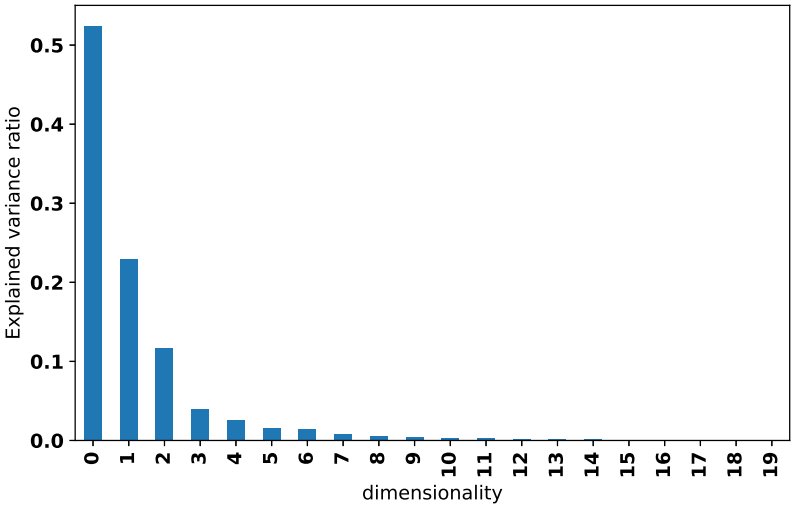
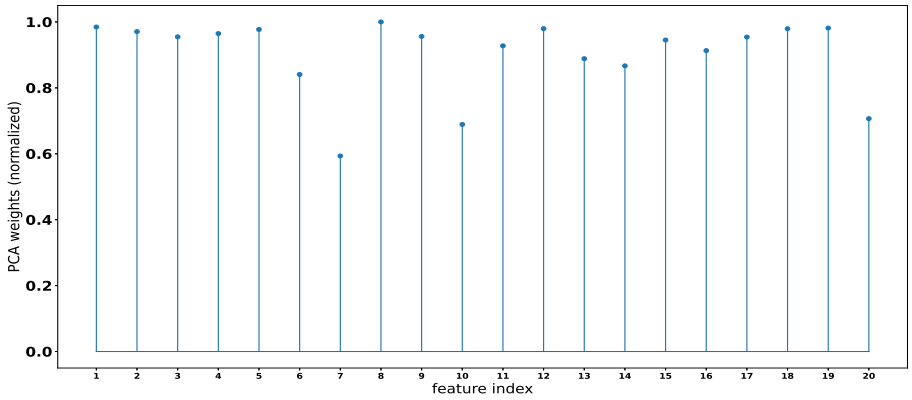


Figure 6.7 – OFB features for axis I

Feature ranking hyperparameters



(a)



(b)

Figure 6.8 – (a) Explained variance ratio (b) PCA weights

In Figure 6.8(a), it can be observed that only 5 out of 20 components constitute of 95% of the variation in the LFB dataset. Further, the resultant PCA weights are visualized in Figure 6.8(b) where the PCA weight correspond to feature importance of that particular feature set. There is high importance given to multiple features with a smooth decrease in the feature weights. This conveys that there are multiple directions of maximum variance in

this feature set and due to this unclear trend it might fail to predict the best feature for this feature set.

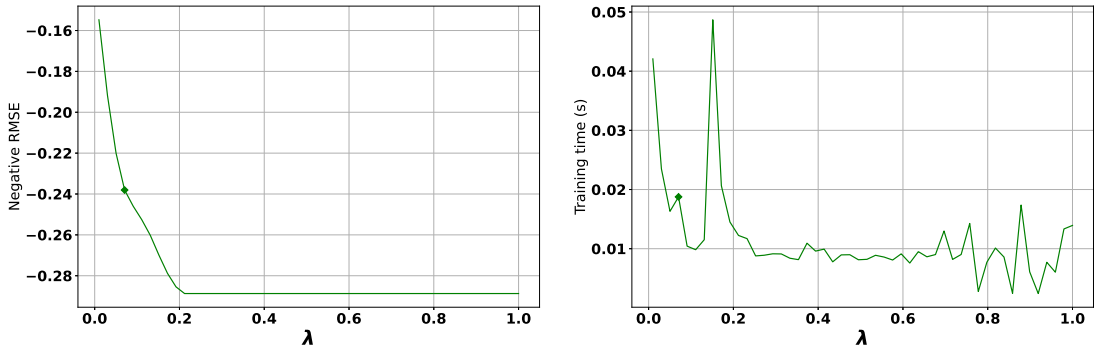


Figure 6.9 – LASSO hyperparameter selection

The hyperparameter scan for the FR is conducted as mentioned in Table 5.1. For LASSO, as displayed in Figure 6.9, the RMSE was chosen as a metric of evaluating the optimal hyperparameter in the defined region. The overall training time decreases with an increase in the value of λ as more value of λ introduces sparsity and chooses less number of features. The scan conducted over all the feature sets showed similar trends. The optimum value for the regularization parameter found was 0.07 which offered an RMSE of 0.24 as it jointly optimizes the training time and performance measures. Also, for very low regularization parameters LASSO would not conduct any FS. However, one drawback of this approach is that it would only choose a small subset of features and therefore only offer reliable results for very low feature dimensionality.

Layer	Number of neurons per layer			
	16	32	64	128
2	0.053	0.13	0.15	0.09
3	0.03	0.076	0.04	0.03

Table 6.1 – Mean RMSE on validation data for CO on LFB dataset over the scan of hyperparameters

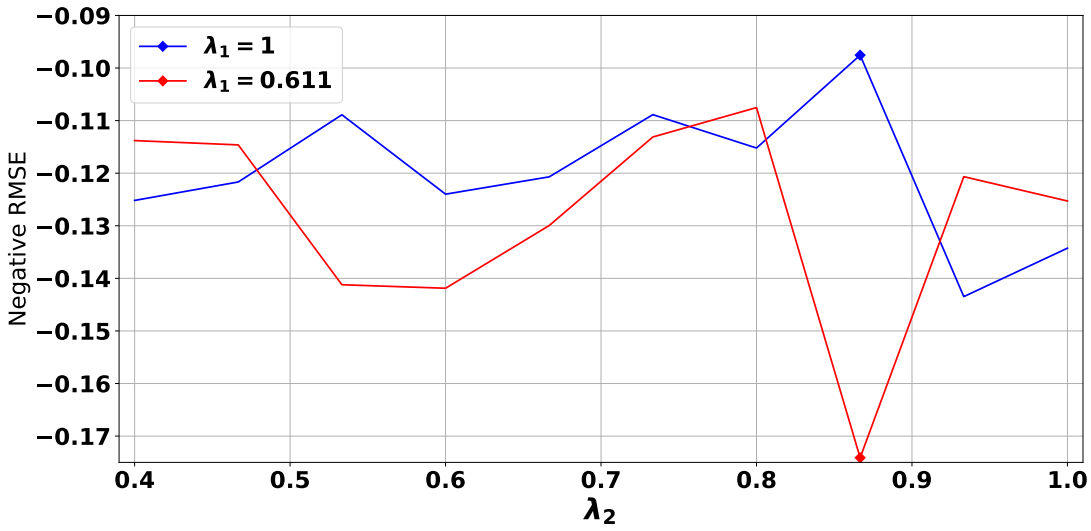


Figure 6.10 – CO hyperparameter selection

For CO, the best model performance characterized by the validation loss for the LFB feature set was found to be a neural network with 3 layers with 16 neurons each for the first two layers followed by a dense layer of unit 1 as displayed in Table 6.1. As described in Figure 6.10, the performance over the validation set has multiple peaks and troughs over the entire scan of the hyperparameters (λ_1 and λ_2). Irrespective of the resolution of the search of the hyperparameters, this trend is observed. The optimal and the worst performances are depicted in the graph with the optimal performance of 0.10 RMSE for a λ_1 value of 1 and λ_2 of 0.867. The worst trough observed is for a λ_1 value of 0.611 and λ_2 of 0.867 with an RMSE of 0.174. The same approach when conducted for the other feature sets, a similar trend is observed as described for the LFB feature set, and the optimal parameters found has a performance measure (validation RMSE) which is very similar to the one from the LFB feature set due to occurrence of multiple peaks. Therefore, the choice of the optimal parameters remains an interesting scope for future work.

The values that correspond to the troughs are to be avoided during conducting CO. One of the reasons for the occurrence of this trough can be due to the occurrence of weights close to zero in the dense layers ahead of the CO layer which inhibits the transmission of data in the forward direction. Another alternative simple solution can also be to simply not include the regularization parameters in the CO loss function. The training time is however very expensive for CO as compared to other FR algorithms investigated for this thesis and also shows a relatively constant performance with mild fluctuations in the scan of the regularization hyperparameters. For the given optimal parameters the corresponding training time is one of the lowest around 10.1 seconds.

The timing performance for all the four FR algorithms in all the four variations of the feature sets is illustrated in Figure 6.11.

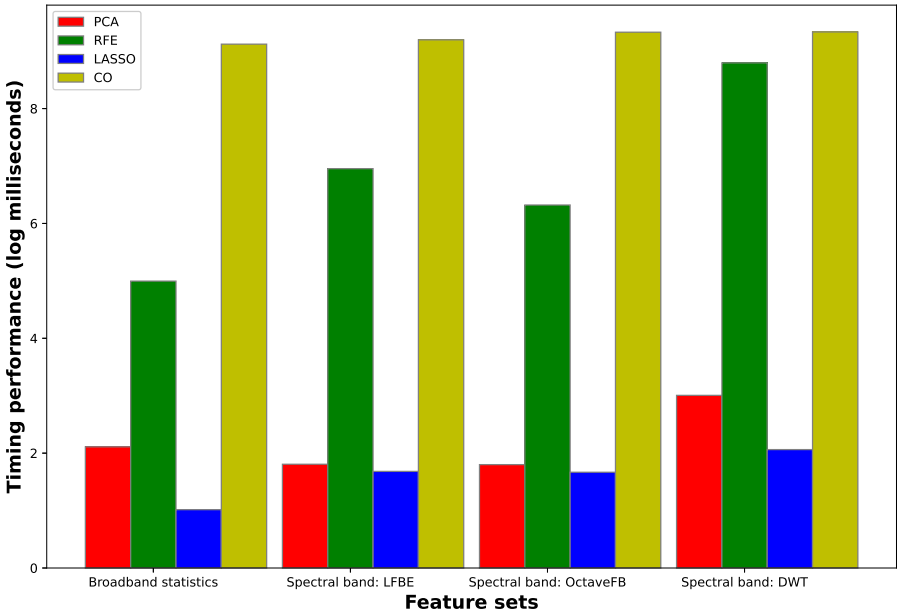


Figure 6.11 – Timing performance analysis of all FR algorithms

The mean time is plotted after taking 10 realizations for each algorithm. The mean time is chosen for evaluation as the standard deviation of the run-times for 10 realizations of all the algorithms was significantly low. Additionally, the plot is visualized in the logarithmic scale measured in milliseconds (ms) for a better visualization. It could be observed that PCA offers the best timing performance for all four feature sets, i.e 6 ms for both LFB and OFB, while CO performs the worst with 11325 ms for the DWT feature set. RFE timing performance is heavily dependent on the dimensionality of the feature set and is inversely related to the dimensionality of the feature set. Furthermore, CO and LASSO have a reasonably constant performance for all the feature sets irrespective of their dimensionality and CO has an overall highest training time.

Linear model performance

After conducting a grid-search with a 10 fold cross-validation over the training data with the LFB features over the hyperparameter list as described in Table 5.2, the best optimizer found is *cholesky* and the regularization value found is 0.01. This setting is used for conducting training over the training data after pre-processing for the linear model and the performance testing discussed below is conducted using the test data after pre-processing. The illustrations below describe the changes in performance with the number of features in the order of FR obtained by the FR algorithms. Negative RMSE is used as a metric for the visualization and

therefore the greater the value, the better is the corresponding performance. Further, the top five features for all the FR algorithms are further described in Appendix A.

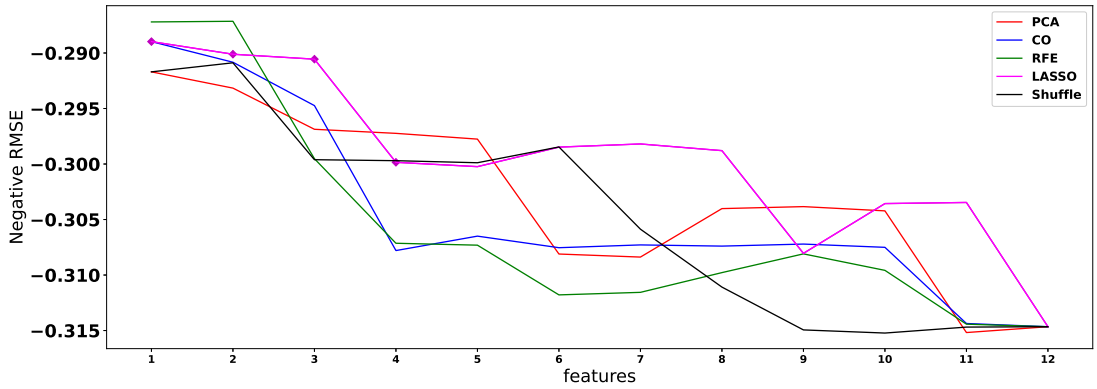


Figure 6.12 – Broadband statistics performance analysis with LR Model

In Figure 6.12, the curse of dimensionality trend is observed very early in the first three features with very little variation in the performance for all four algorithms. RFE gives optimal performance with two features with an RMSE of 0.287. The two features evaluated by RFE are SF from both the accelerometer axis. The obtained features have variations coming out of all four FR algorithms. RFE evaluates two features that come from the first axis (SF and RMS). Another interesting observation is that LASSO has the best training time of 2 ms with an RMSE of 0.289 (which is very close to the performance of RFE) with a single feature that is the variance from the first axis. However, LASSO only selects 4 useful features that are ranked in the figure. This is due to the shrinkage of other values to 0 due to regularization. Further, PCA with a training time of 6ms has its optimal performance of an RMSE value 0.292 with a single feature (second axis Kurtosis). Overall, there is an agreement of 2 out of 4 features of PCA in the top 5 ranks with LASSO with Kurtosis from both axis being a common feature. CO has an optimal performance of 0.288 with the best three features as RMS from the second axis and variance from both the axis. Therefore overall, RFE would be ideal for system run-time performance as it offers very good model performance for this feature set with only one feature. Here, as the number of features increase sometimes the FR algorithms fail to identify useful features as to random shuffle. This is because of the addition of low ranked features with the increment of feature dimensionality, the model tends to overfit. While this is not applicable to random shuffle as there is no FR strategy involved.

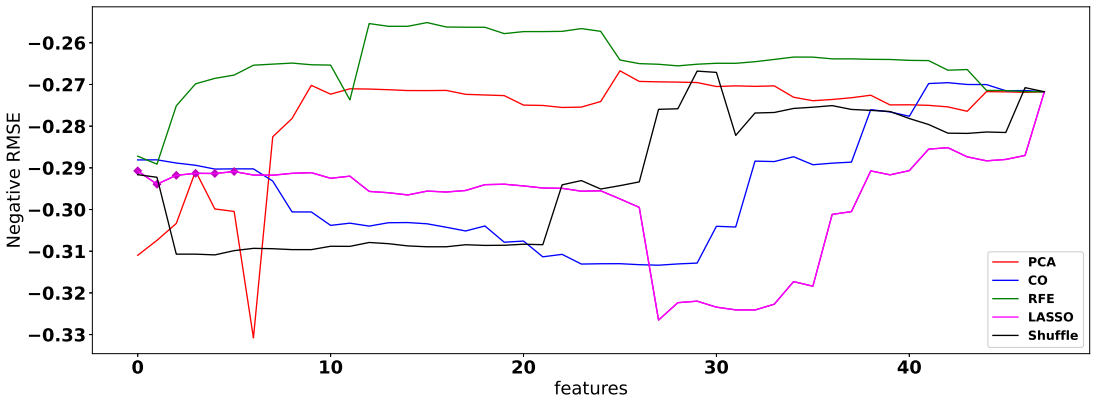


Figure 6.13 – DWT features performance analysis with LR Model

The performance analysis of DWT features is conducted similarly and is illustrated in Figure 6.13. The best performance offered is by RFE with a minimum RMSE of 0.255 with a total of 15 features. Here, out of 15 features, 10 come from the first axis. The RMS and SF have the highest amount of importance for all the bands from both axes. The addition of the second band RMS brings an improvement of 0.01 RMSE as the addition of the third feature while the addition of the thirteenth feature as the variance from the second axis (second band) offers an improvement of 0.02 RMSE (for RFE). There were no features chosen from the second axis with the lowest DWT subband. Through a system design lens, this could result in a reduction of an axis (the second axis here due to the occurrence of the most features from the first axis) would offer a reduction on one DWT band resolution and thereby resulting in a 25% boost (1 out of 4 dwt bands reduced at pre-processing) in the number of samples at the pre-processing part of the pipeline and an additional lowering of hardware cost by reduction of an accelerometer axis. Interestingly, PCA obtains 10 optimal features with an RMSE of 0.272. Thereby, PCA has a decline of 5% in model performance as compared to RFE without any exposure to a label during its conduction. However, the findings of PCA are different from RFE due to its reliance on the explained variance ratios in the description of the feature set. For PCA, there are only 6 out of 10 features that come from the first axis. Also, 9 out of 10 features come from the first three subbands of the DWT and the majority of them come from the first axis. The addition of the eighth feature brings an improvement in performance from an RMSE of 0.33 to 0.28. This feature is the RMS value from the first axis second dwt band. Again, from a system design point of view, a reduction of the last dwt bands could be conducted. Further, the curse of dimensionality trend is observed at around 15 features for all algorithms thereby making FR viable for this feature set which has the highest dimensionality. The addition of all features offer an RMSE of 0.27 for this feature set and RFE offers a reduction of 33 features with an improvement in performance. Additionally, LASSO and CO characterize the most features which are noisy

for this feature set with LASSO selecting upto 5 features that do not add any perfomance addition.

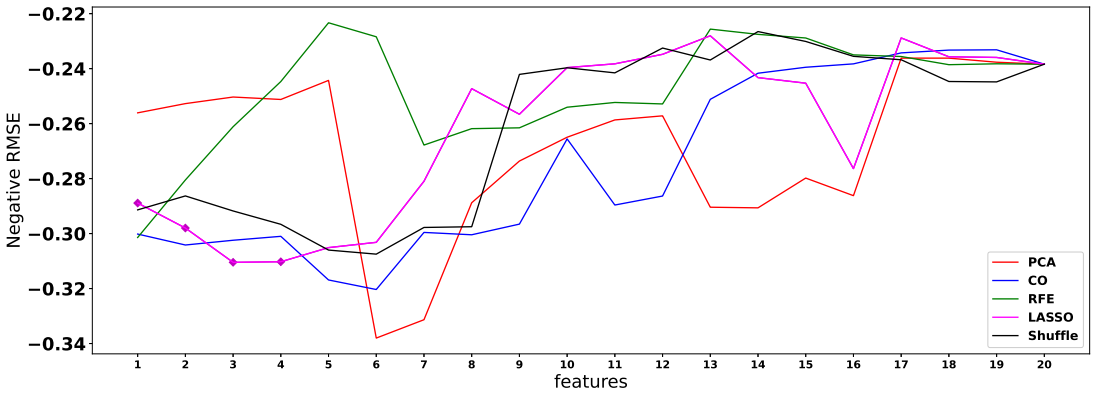


Figure 6.14 – LFB features performance analysis with LR Model

In Figure 6.14, the performance analysis of the LFB feature set is visualized. RFE offers the best RMSE value of 0.223 with only 5 features. Three out of five features come from the second axis with a high bias from the features obtained from the higher parts of the frequency spectrum. PCA offers an RMSE of 0.25 with only 5 features where three come from the second axis. The addition of the sixth feature as the fifth band power from the first axis brings a decline in performance from an RMSE value of 0.24 to 0.34. For this feature set, the curse of dimensionality trend is observed from around 6 features for all the FR algorithms. LASSO fails to find out the optimal features for this feature set that are calculated by RFE and is only reliable up to 4 features and its performance is poorer than a random shuffle for this feature set except for the first feature. Finally, this feature set having relatively low dimensionality overall offers the best model performance out of all the feature sets.

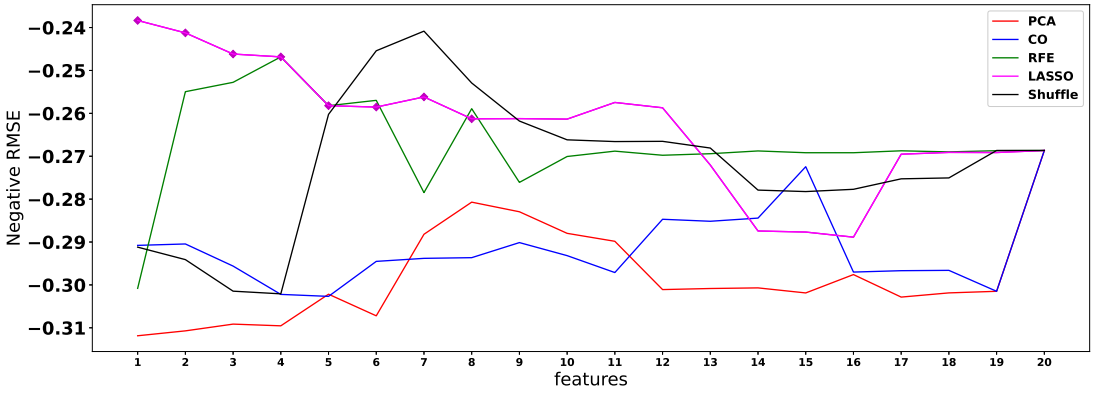


Figure 6.15 – OFB features performance analysis with LR Model

For the OFB feature set, the model performances are displayed in Figure 6.15. LASSO offers an optimal performance of an RMSE of 0.24 with only one feature. This is the ninth band from the first axis. Further, LASSO selects 8 features that are the highest out of all the other feature sets. An interesting trend could be observed for this feature set as the random shuffling of features when pushed into a model offers a similar model performance with an RMSE of 0.246 with 7 features. Here 5 out of 7 features come from the first axis with a high bias from the higher regions of the spectrum. Again, FR sometimes fails to compute the useful features for higher feature dimensionality. Also, RFE has a similar performance with 4 features evaluating into an RMSE of 0.247 with a training time of 556 ms, having only a 0.04% decline in model performance as to random shuffle. Further, two out of the last three features obtained by the random shuffle that contributes to performance are also calculated by RFE. The only important feature that RFE failed to compute is the seventh band power from the first axis. Thereby, FR could still be a viable option always as to random shuffle.

Non-linear model evaluation

Layer	Number of neurons per layer			
	16	32	64	128
3	0.12	0.10	0.09	0.09
4	0.1	0.09	0.09	0.1

Table 6.2 – Mean RMSE on validation data for PRONOSTIA dataset over the scan of hyperparameters

For the evaluation of the design of MLP, the performance is evaluated around the twelfth feature where the curse of dimensionality trend is observed in the LFB feature set on the RFE FR indices. An MLP on the chosen hyperparameters of a 3 layer (2 hidden layers and one dense layer) with 64 neurons MLP offers optimal mean performance (with very low variance) as displayed in Table 6.2 and the performance is discussed after conducting a Monte Carlo simulation of 10 iterations. This setting of the hyperparameters is chosen because there are no accelerating returns in model performance with the increase in the number of model parameters and high model complexity is suboptimal due to high training times in the final wrapper. The variance in the performance metric is in the order of 10^{-5} during the performance of all feature sets.

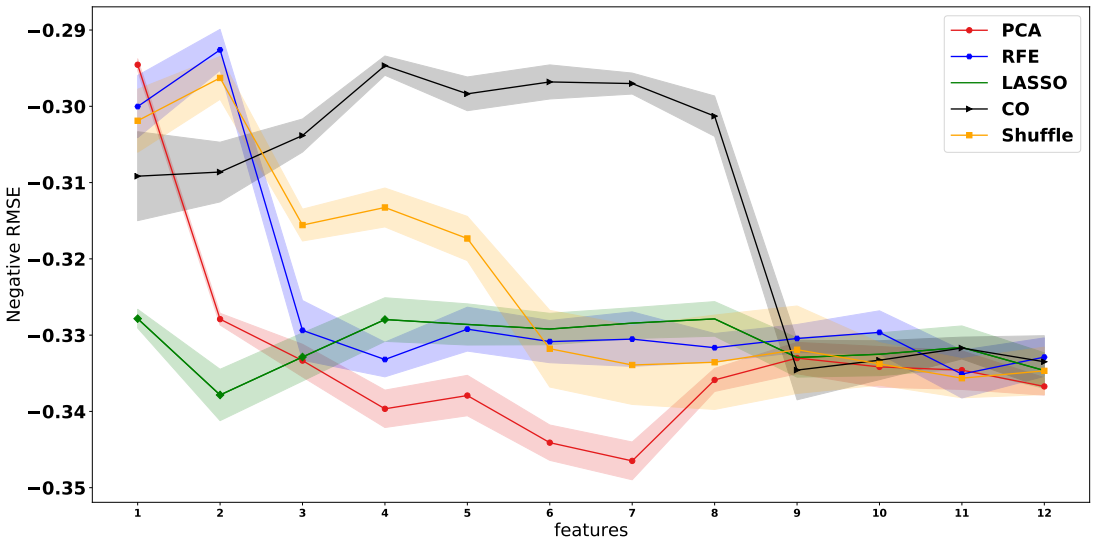


Figure 6.16 – Broadband features performance analysis with MLP Model

In Figure 6.16, the trend in the performance is very similar to that of the corresponding linear model. The curse of dimensionality happens early after the addition of three features where the performance evaluated by the top three features of CO offers an RMSE value of 0.30. The addition of the second feature in PCA (second axis Kurtosis) corresponds to a large dip in performance of mean RMSE from 0.294 to 0.33 as opposed to correponding performance dip in the linear model which offers an RMSE of 0.295. CO has an increasing trend with its optimal performance of 0.29 RMSE at 4 features. The variance in the performance metric is lowest for PCA for the first two features around 10^{-6}. The overall variance is therefore low for the measurement of the performance of RMSE. However, the overall best performance when wrapped with the same set of feature subsets (obtained by the FR algorithms) offers better performance for the linear model for this feature set.

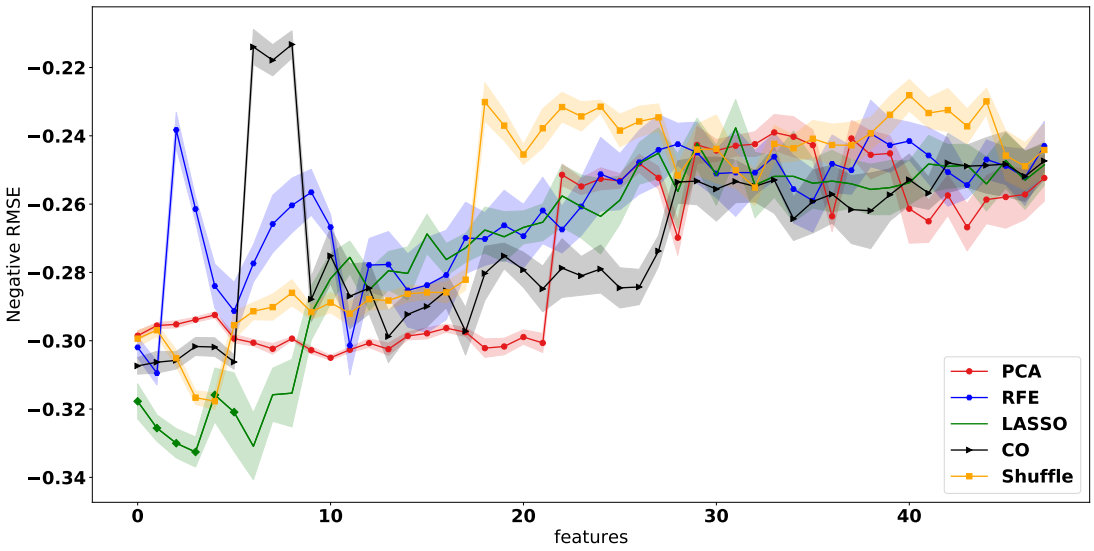


Figure 6.17 – DWT features performance analysis with MLP Model

In Figure 6.17, the overall trend is again very similar to the linear model. The best performance is offered by CO for this feature set with a mean RMSE of 0.21 with 7 features only. Here 5 out of 7 features come from the first axis information which can result in a reduction of sensor axis. Here, 6 out of 7 features occur as the Kurtosis and CF statistics from different bands. Additionally, RMS has an incremental trend in the last few lifecycles of the bearing while the CF has peaky trends throughout the machine life. The performance offered by RFE for the third feature rank is an RMSE of 0.24, which is lower than the linear model which offers around 0.27. However, for RFE the inclusion of the fourth feature brings a decline in model performance for MLP as opposed to the linear model. The increment in the addition of the thirteenth feature is however similar for both models. After the thirteenth feature, both the MLP and the linear model performance stagnate. PCA offers an overall best performance for MLP with an RMSE value of 0.24 with 34 features which is better than that of the linear model. Here, 23 out of 31 features originate from the second axis information. The addition of the twenty-seventh feature offers a major improvement in performance from an RMSE of 0.30 to 0.22 and this feature is the CF from the last DWT band from the first axis. Overall, the MLP offers better performance for this feature set as compared to the linear model and other feature sets with the only drawback of mild uncertainties involved in the prediction.

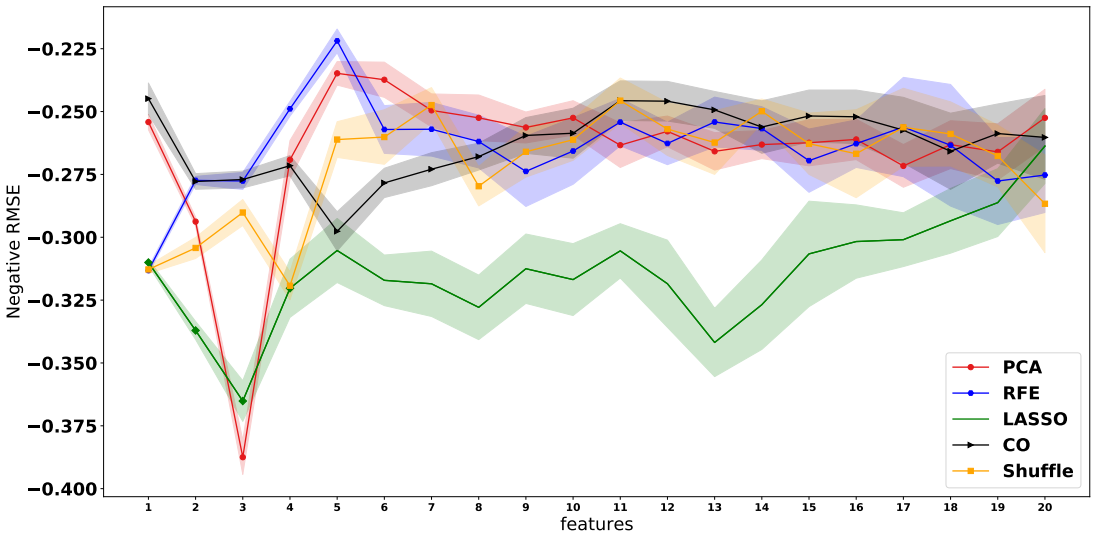


Figure 6.18 – LFB features performance analysis with MLP Model

In Figure 6.18, the trend is very similar for all four algorithms. From the sixth feature onward the MLP represents all the remaining features as noisy, unlike the linear model that does from the ninth feature onward. The best performance offered is by RFE an RMSE of 0.221 with fifth features and is very similar to the performance of the linear model with the same number of features. Further, there is a dip in performance for both LASSO and PCA for the addition of the second and third feature from an RMSE value of 0.25 to 0.39 for PCA and from 0.31 to 0.37 in LASSO. The correponding features come from the first (axis I) and the ninth (axis II) LFB band for PCA and third (axis I) and seventh (axis I) band for LASSO. Overall, PCA offers the best perfromance value of 0.23 RMSE with the top five features.

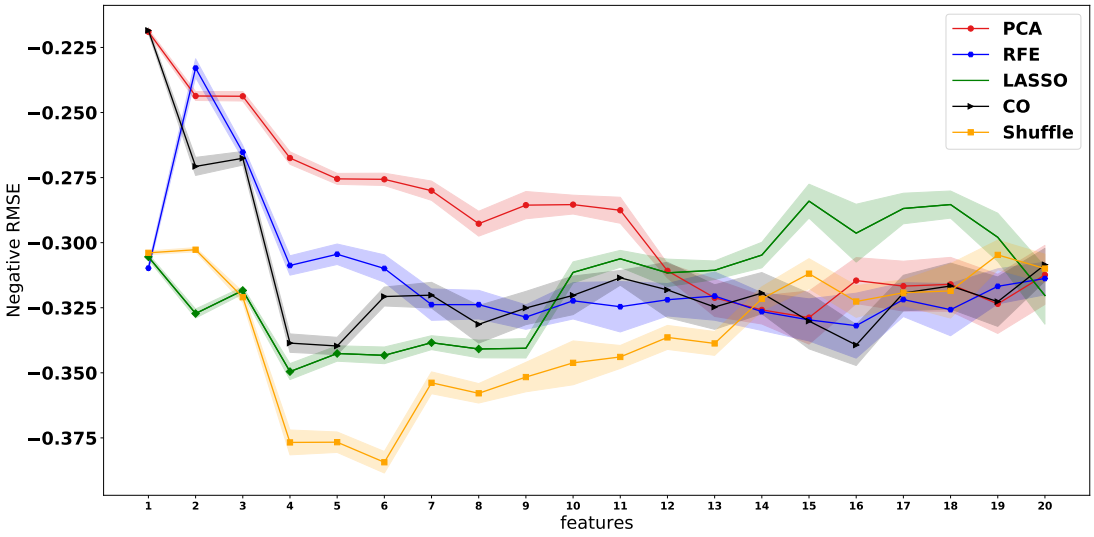


Figure 6.19 – OFB features performance analysis with MLP Model

In Figure 6.19, the best performance offered is by PCA and CO with a single feature offering an RMSE of 0.22. The PCA algorithm computes the eighth band from the first axis information while CO computes the first band from the second axis as the most important feature. The curse of dimensionality trend is observed from six features with an RMSE value of 0.28 for PCA. Lastly, the random shuffle has extremely poor performance for this feature set except calculating the second feature where it surpasses LASSO. This feature is the sixth band information from the second axis. Overall, FR is viable for the PRONOSTIA scenario.

Feature set	Linear Regression	MLP (mean score)
Broadband statistics	0.315	0.334
DWT statistics	0.27	0.248
LFB	0.24	0.26
OFB	0.27	0.32

Table 6.3 – RMSE score of feature sets with all features (no FR)

The RMSE scores of the linear and non-linear models (approx.) without for all feature sets are given in Table 6.3. These are performance scores when all features are pushed into the model and are used to evaluate the contrast in how much improvement occured with the introduction of FR.

In summary, the PRONOSTIA scenario having a total number of 100 features, the best performance from the overall scenario arrives from the DWT feature set in a non-linear model with only 6 features by CO. This ensures a reduction of 42 features in the number of features from the DWT feature set for this problem with a trade-off in model complexity. For the linear model, LASSO offers the best performance with a single feature for the OFB feature set and RFE for the LFB feature set with only 5 features. Further, LASSO can be only useful for investigating performances with scenarios that require extremely low amount of features. This is due to it's nature of thresholding other indices to zero. Further, there is more reliablity in PCA, LASSO and RFE in terms of repeatability due to the relatively low set of hyperparameters. However, CO is more unreliable due to the large amount of hyperparameters involved and the stochastic nature of neural networks. But, CO is still viable in feature rejection (ranking noisy features reliably). In addition to that, if CO is used for model training, it is vital to tune the regularization parameters due to its oscillatory trend.

Furthermore, a broadband analysis might not be sufficient for the PRONOSTIA scenario as the spectral features can offer a significant improvement from a system design perspective. Finally, the MLP offers a better performance in the spectral analysis with a better representation of noisy features for both broadband and spectral features.

6.1.2 FAN

For the FAN dataset, the LFB features have shown promising performance with a linear model and are therefore chosen for investigation in conjunction with the FR methodology. The F1 score for the broadband, DWT, and OFB are 0.16, 0.5, and 0.17 while the LFB feature set had a score of 0.7. A higher F1 score contributes to a better performance. Further feature analysis is conducted below using the proposed methodology to investigate if FR could improve the performance for this scenario with less number of features. Firstly, the feature set is visualized and then the results of the methodology are discussed for both the linear and nonlinear models.

Feature extractor

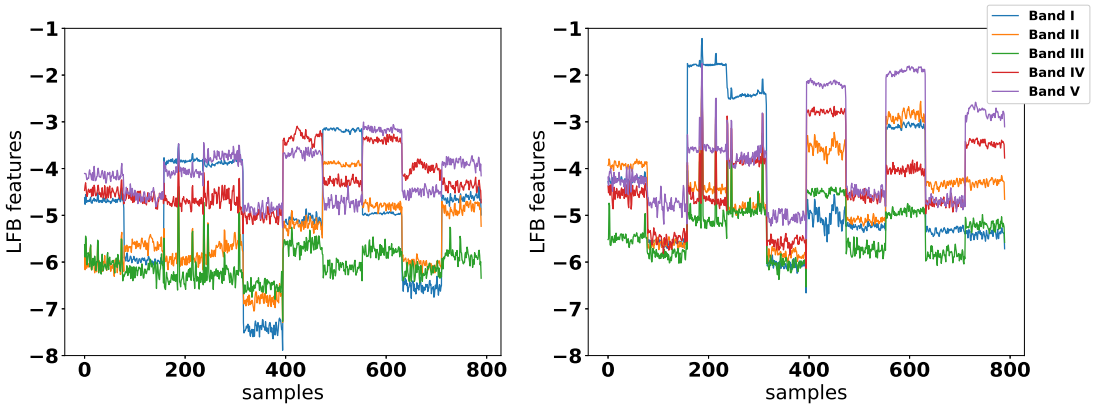


Figure 6.20 – LFB features for FAN Dataset in good condition

The LFB features for the good condition fans are visualized for two axis in Figure 6.20, a combination of step function like response is observed in the feature. The variation in the data observed is highest for the lowpass filter bank (Band I), while the high frequency components have lower dynamic range.

In Figure 6.21, the LFB features are visualized for the fans with a bearing defect and the variance is high for the highpass filterbank and the step function like responses are more distorted for both the hardware axis.

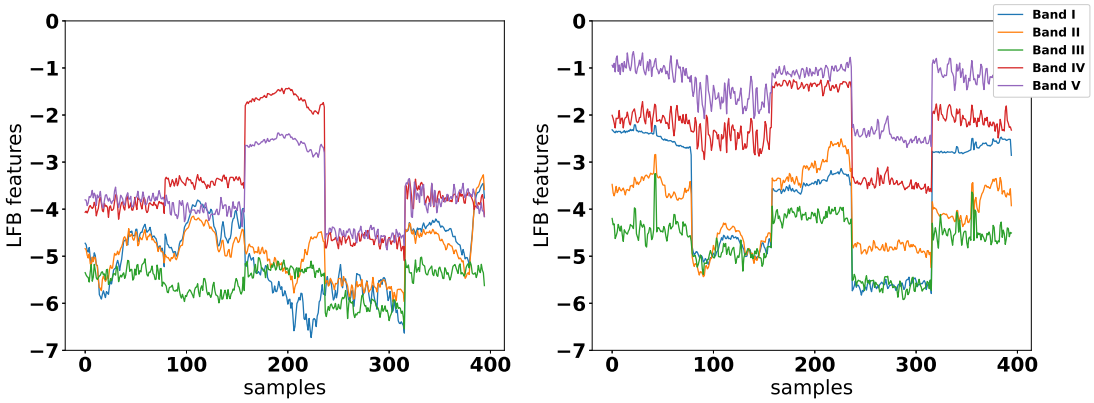
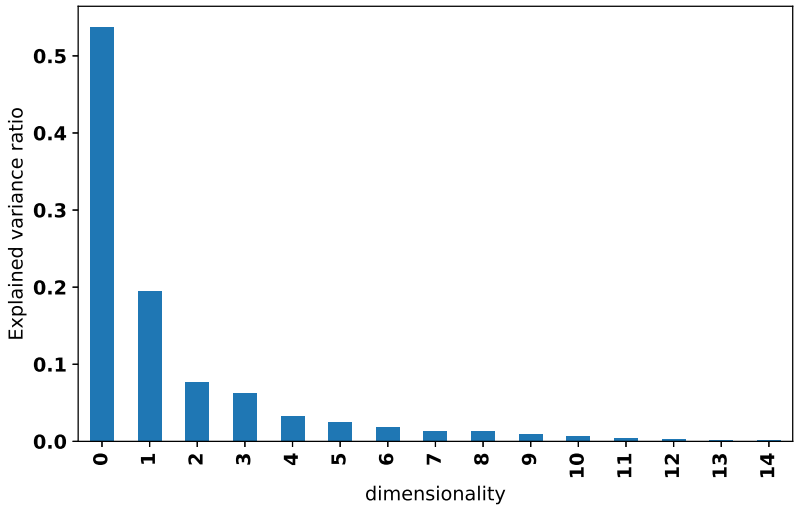
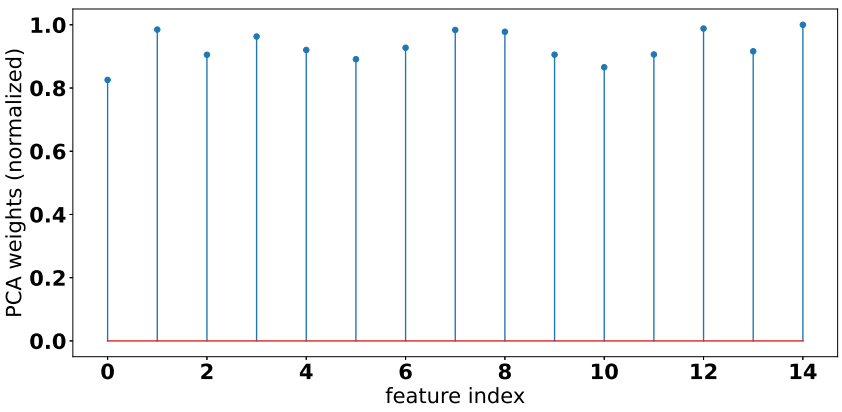


Figure 6.21 – LFB features for FAN Dataset with Bearing defect

Feature ranking hyperparameters



(a)



(b)

Figure 6.22 – (a) Explained variance ratio (b) PCA weights

It can be observed in Figure 6.22(a) that 95% of the variance of the data can be explained with only 7 out of 15 dimensions for the FAN scenario. Further, in Figure 6.22(b) the normalized PCA weights are visualized which correspond to feature importance for this FR method and high differences in the weights are not observed. This shows that variation in the fan dataset is not dominated in one particular direction in the feature-space. This conveys upto an extent that this might not be a very good predictor of FR.

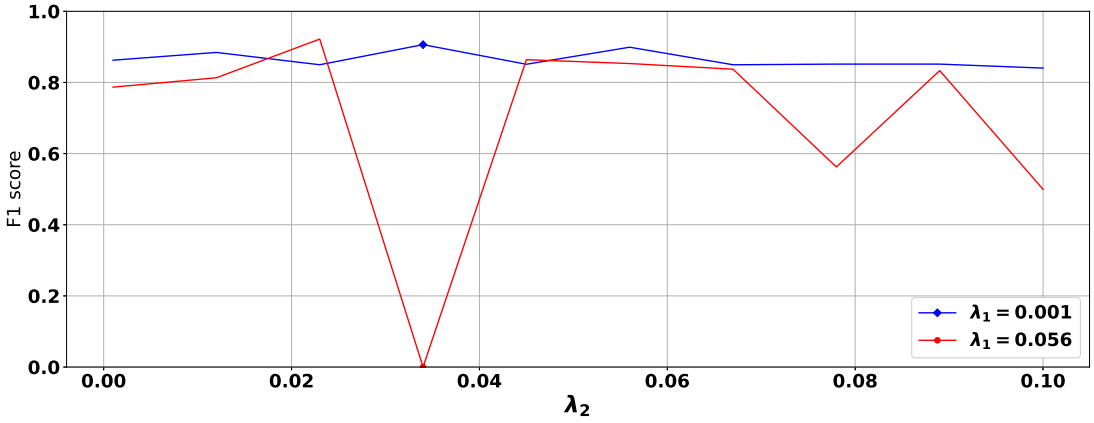


Figure 6.23 – CO hyperparameter selection

For CO, the best model performance characterized by the validation loss was found to be a neural network with 2 layers with 16 neurons each for the first layer followed by a dense layer of unit 3 with softmax activation function. For this chosen setting the value of the F1 score on the validation data was very close to 1 and larger model parameters also contributed to a similar score. However, hyper-parameter tuning with different proportions of train and validation data can be an interesting scope of future work. As described in Figure 6.23, again the performance over the validation set has multiple peaks and troughs over the entire scan of the hyperparameters (λ_1 and λ_2). The optimal and the worst performances are depicted in the graph with the optimal performance of 0.95 F1 score for a λ_1 value of 0.001 and λ_2 of 0.034. The worst trough observed is for a λ_1 value of 0.056 and λ_2 of 0.034 with an F1 score of 0.

These values that correspond to these dips are to be avoided during conducting CO. Again, one of the reasons for this dip can be due to the weights of the network after the CO layer become very small and hence the input features are unable to move forward in the network. The training time is however expensive for CO and shows a constant performance in the scan of the hyperparameters. For the given optimal parameters the corresponding training time is one of the lowest around 10.8 seconds.

The timing performance for the three FR algorithms are 0.04 seconds for PCA, 10.82 seconds for CO and 15.3 seconds for RFE.

Linear model evaluation

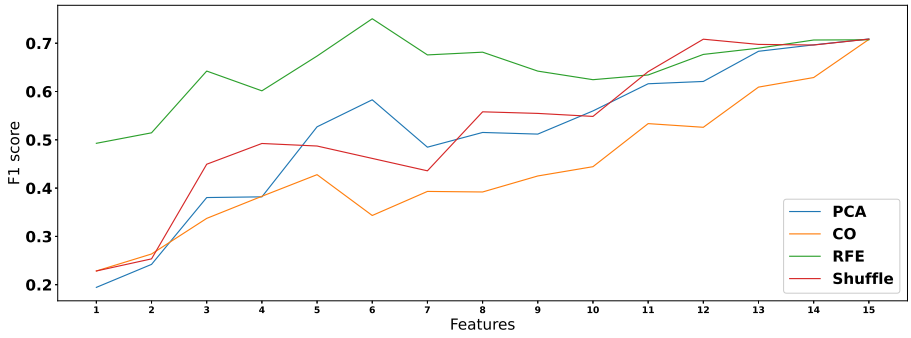


Figure 6.24 – LFB features performance analysis with LR Model

After conducting a grid-search with a 10 fold cross-validation over the training data with the LFB features over the hyperparameter list as described in Table 5.2, the best optimizer found is *liblinear* and the regularization value found is 1.

In Figure 6.24, the linear model performance is visualized. RFE performs best with only 6 features originating from the first two accelerometer axis with an F1 score of 0.75. This confirms a reduction of hardware cost by one accelerometer axis and a reduction of several features by 60%. The curse of dimensionality trend is however not observed by all the other FR algorithms except RFE. PCA fails to obtain good performance for low feature dimensionality due to a lack of variation in the PCA weights. CO shows an increasing trend till the fifth feature with a sudden drop and again a constant performance improvement. All models converge to an F1 score of 0.71. Therefore, for a deployment system performance RFE can be a good candidate for conducting FR which obtains a 5% improvement in performance as compared to the performance evaluated by all features.

Non-linear model evaluation

The non-linear wrapper chosen for this scenario is a 3 layer MLP with two hidden layers with 16 neurons per layer followed by a dense layer of 3 units with a softmax activation function. This is because there is a constant performance in terms of the F1 score on the validation set for the MLP with an addition of further hidden layers. The performance on the validation set has been very high and almost close to 1 for this setting. Therefore, this setting can offer minimal model complexity and faster training times.

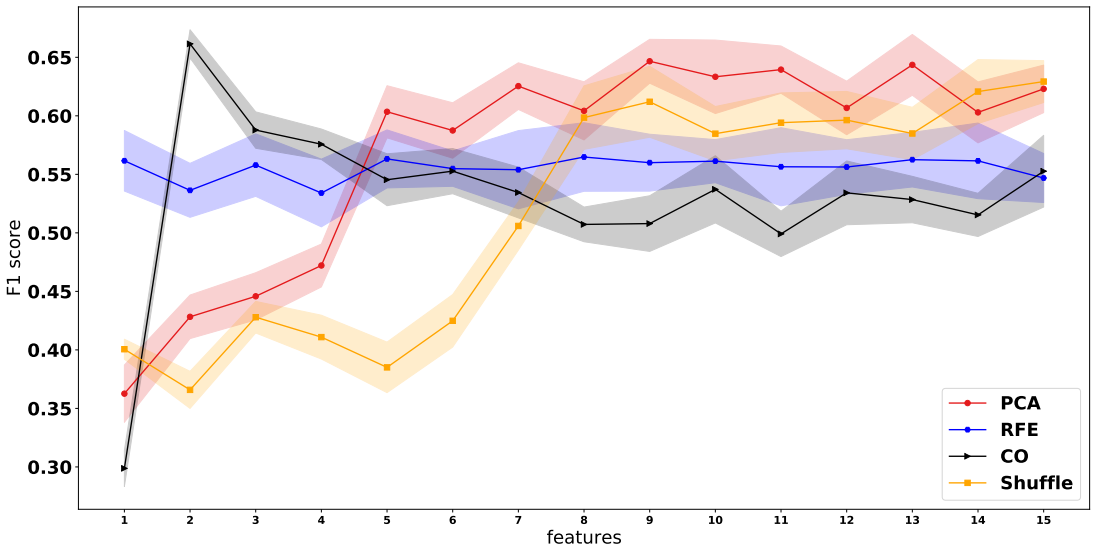


Figure 6.25 – LFB features performance analysis with MLP

In Figure 6.25, the performance of the LFB feature set is visualized with the MLP wrapper. The variance of the obtained values is high and is in the order of 10^{-3}. The best performance is offered by PCA with 8 features with an F1 score of 0.66. However, CO offers an F1 score of 0.66 with two features. The feature obtained by CO are the third axis last two band information which confirms a reduction of hardware cost by two accelerometer axis with the trade-off being high training time for CO. However, the findings of RFE do not agree with that of CO as RFE finds all the features from the first two axis information. This shows that the appropriate choice of the final wrapper is important. Additionally, four out of six features in the PCA algorithm comes from the first two axis. Finally, a random shuffle doesn't compute the optimal features that contribute to model performance for low feature dimensionality making FR viable for this scenario (except for high number of features).

The performance of all features (without FR) on the linear model on the LFB feature set leads to an F1 score of 0.70 and a high variance In summary, for this scenario, the linear model offers better performance with RFE and for the non-linear model with a reduction of hardware cost by an accelerometer axis (2 out of 3 offers performance) for CO with tradeoffs with model complexity. However, CO has lower reliability due to the high range of hyperparameters involved and the stochastic nature of neural networks. Therefore, RFE is the optimal method in the given set of FR algorithms due to it high performance of 0.75 with only 6 features and its reliability. Overall, there is more agreement in the choice of features occuring from the first two axis.

6.2 Discussion

A further discussion of the obtained results is carried in this section first for the PRONOSTIA scenario followed by the fan dataset.

PRONOSTIA

An analysis of the results obtained in Section 6.1 are discussed here for all the FR algorithms and their limitations followed by the limitations of the final wrapper and the metric involved during evaluation.

RFE brings an overall performance improvement with a reduction of 11 features for the linear model performance for the broadband statistics, 33 features for the DWT feature set, and 15 features for the LFB feature set. SF and RMS are two of the high-ranked features from both the broadband and the DWT feature sets. However, RFE FR indices when wrapped to an MLP for the DWT feature set, the performance improved with only two features but it failed to find out the optimal performance when compared with the other FR algorithms. This can be due to the overfitting of the MLP model. However, for the OFB feature set, RFE fails to obtain optimal performance. There can be two reasons leading to this result. Firstly, a linear regression wrapper can be one of the limitations of this FR algorithm. This is because the linear regression coefficients are the only metric to obtain FR for this algorithm. Secondly, the OFB feature set has a high variance in the features as observed in Figure 6.7 and further pre-processing might be necessary to obtain a better representation of this feature set. This is because most of the features obtained from this feature set are from the high bands of the spectrum. However, due to the nature of the design of the OFB, the frequency resolution for higher parts of the spectrum is poor. Lastly, the training time of RFE scales incrementally with the dimensionality of the feature sets and therefore an introduction of a model inside RFE involving high training times can be an overhead to this algorithm.

For PCA, a similar trend is observed when the FR indices on the OFB feature set are pushed for the linear model, a degradation of performance is observed as compared to the performance observed by the other FR algorithms. This can be because of two possible reasons. Firstly, the 95% explained variance ratio considered for this algorithm might include some noisy representation of the given feature set. Since PCA is completely dependent on the input features without the exposure of the label, the optimal choice of the explained variance ratio can be different for different feature sets leading to one of the limitations of this algorithm. However, the FR indices obtained in this algorithm offer better performance for all the other feature sets with having 3 common features with RFE (offers best performance on the final wrapper) for the LFB feature set and 2 common feature for the broadband statistics.

For LASSO, better performance is obtained with the OFB feature set. One of the reasons leading to this result can be due to the nature of thresholding that is involved in the LASSO algorithm. The thresholding operation leads to a rejection of noisy features. However, the limitation of thresholding is that it only selects a very small subset of features and therefore an analysis with this algorithm would be restricted to small feature dimensionality. In addition to that, with lower values of regularization, FS will not be introduced to this algorithm. Overall, this FR method requires very low training time and is one of the benefits of this approach.

For CO, one of the drawbacks is the fine-tuning of the regularization parameters involved as it shows oscillatory behavior. This is because in addition to avoiding the troughs, the peaks obtained throughout are multiple and therefore a heuristic approach to obtaining the optimal parameters might be one of its limitations. In addition to that, the intermediate size of the model in CO is directly responsible for its performance and training time.

For the final wrapper, it can be observed that when the test feature set is evaluated with the same FR indices there is a difference in model performance for linear and non-linear models. For example, it can be observed that for the broadband feature set, both the linear and the non-linear models add diminishing returns with any addition of features after the first three features. One of the reasons could be overfitting with the introduction of too many features. This is one of the limitations of this methodology as a careful evaluation of hyperparameters is required while choosing the final wrapper. The trend is observed more in the MLP and therefore more focus is required in the fine-tuning of hyperparameters for non-linear models. However, the goal for this thesis is restricted to investigating how much performance improvement could be obtained with the least number of features. In addition to that, due to the stochastic nature of the MLP, there is a lack of reliability in terms of convergence. Convergence in the MLP can be described as the progression towards a state of the MLP parameters where the MLP has learned to properly respond to a set of training data within some margin of uncertainity. This is however more reliable when it comes to linear models due to better control over its hyperparameters. Lastly, RMSE can be used as a metric for evaluating the performance for this scenario as it offers large errors to inaccurate predictions. However, one limitation of RMSE is that it reflects early and late predictions of the RUL value equally. Therefore, the choice of the metric might need to be varied with the requirements of the scenario if early and late predictions translate to different penalties in system deployment performance.

FAN dataset

An analysis of the results obtained in Section 6.1 is discussed here for all the FR algorithms for the fan dataset followed by a discussion on the limitations of the final wrapper and the metric involved in performance evaluation.

For RFE, the FR indices on the LFB feature set lead to the best performance for the linear model. This is however not observed when the same FR indices on the LFB feature set are wrapped around an MLP model. One of the reasons leading to this result might be due to the fact that RFE has a logistic regression wrapper whose weights contributing to FR metrics help represent the order of FR which would be more reliable for a linear model. This brings to another possible limitation of this methodology while selecting a wrapper or embedded FR method, as the model that performs the FR can compute an order of features that might better fit a certain class of models that are used for the final wrapper. However, the choice of a logistic regression model in the RFE wrapper is motivated by lowering the training time of this FR algorithm and the model used in the final wrapper has an l1 penalty involved leading to different models and thereby making the comparison viable. In addition, RFE (and all the other FR algorithms) has no exposure to the test data on which the performance is evaluated for the final wrapper.

For PCA, the FR indices when pushed to both the models for the LFB feature set failed to find the optimal set of features that are computed by RFE and CO for the corresponding linear and non-linear models. This can be due to the nature of the input features having a large amount of variance as described in Figure 6.20 and the PCA algorithm has no exposure to the corresponding labels of these features during the training process. Therefore, it has to rely completely on the explained variance ratio as described in Figure 6.22 and the two principal components contribute to almost 73% of the explained variance ratios. Therefore, the introduction of further components can introduce a more noisy representation of the data. This can be one of the limitations of PCA for this methodology for the fan dataset.

For CO, the fine-tuning of the regularization parameters are of vital importance as the troughs displayed in Figure 6.23 offer very low F1 scores and the peaks obtained are multiple throughout the hyperparameter search. Therefore, CO is more sensitive to classification problems. However, the FR indices obtained by CO offer the best performance when evaluated using an MLP wrapper with the LFB feature set with only two features. One of the reasons leading to this result can be because CO being an embedded method with a small-sized MLP wrapped around it for prediction of the FR has learned with a better ability to rank features that can fit better in non-linear models.

For the final wrappers, the non-linear model performs poorly on the evaluation of the FR algorithms. One of the reasons can be overfitting despite the low amount of parameters involved in the MLP wrapper. Further, the LFB feature set lacks a strong variation in the trend in the three classes and can require further preprocessing for the improvement in the performance. Lastly, for the choice of the metric, the macro averaged F1 score can be used as a metric for evaluation as it is sensitive to class imbalance and the occurrence of FN and FP. However, an F1 score would treat both FN and FP with an equal weightage and this can be completely scenario specific.

Chapter 7

Conclusion

In this thesis, a comparison of FR methods for PdM scenarios is studied. The main goal in doing so is to establish a methodology to compare different FR algorithms that are optimized for deployment performance. This is approached by first narrowing down the information processing pipeline to SOTA feature extractors, FR algorithms, and finally a common ML wrapper which is used as a tool for comparing the FR algorithms with one common metric. The hyperparameters of the SOTA FR algorithms are optimized to obtain low training times. A subset of the feature sets is arranged in the order of the FR obtained from the optimized FR algorithms are used in the evaluation of a performance score by pushing them into a final wrapper (a linear model and a non-linear model). A grid-search methodology is used in obtaining the choice of the hyperparameters for the final wrapper.

First, the methodology is investigated on the PRONOSTIA dataset using broadband and narrowband feature extraction over available raw-time series data from two sensor axes from three different conditions. Four kinds of feature extractors are used to extract features from the two sensor axis. These are primarily extracted from the available broadband data and the spectral band information obtained from narrowband in the frequency spectrum of the data. This results in four feature sets: broadband statistics, DWT statistics, LFB power, and OFB power with DWT statistics having the highest feature dimensionality. Further, the FR algorithms are investigated arrive from three different categories: filter methods, wrapper methods, and embedded methods. Four FR algorithms are selected for investigation: PCA (filter), RFE (wrapper), LASSO (embedded), and CO (embedded). Due to the high amounts of training time offered by RFE, a linear wrapper is fed to reduce its training time. For LASSO and CO, the performance of these FR algorithms and the training times on the validation set becomes a metric for the tuning of its hyperparameters. The choice of the optimal hyperparameters of CO is conducted using a grid search strategy, first for tuning the model size and then the tuning of the regularization parameters. The results for this scan show an oscillatory behavior in the performance of CO with the variation in the regularization parameters. Further, a grid search is conducted over an array of optimizers,

model parameters, and regularization parameters to obtain the optimal hyperparameters of the final wrapper. There are two types of models investigated for the comparison: linear and non-linear. Finally, the feature subsets for every feature set are arranged in the order of FR obtained by the FR algorithms are chosen with RMSE as the metric for evaluation, and are evaluated on the test feature set. The LFB feature set provides promising results when tested over a linear model with an RMSE of 0.223 with only 5 features and is achieved through RFE. When tested over an MLP, it offers an RMSE of 0.21 over the DWT feature set with only 6 features through CO. LASSO and PCA offer low training times while the training time of RFE is high and is dependent on the dimensionality of the feature set. However, a drawback of LASSO in conjunction with FR is that it is only reliable for small feature set dimensionality due to thresholding. The methodology is further investigated with a computer fan dataset with time-series data collected from three classes of fans from 3 different sensor axes. After feature extraction, the FR algorithms are conducted over the LFB feature set and the linear model offers the best performance with an F1 score of 0.76 on the indices of RFE on the test feature set. There are variations observed in the order of FR obtained by the two methods.

The scope of future work can be conducted in three different areas in conjunction with the improvement of this current methodology: feature extractors, FR algorithms, and the choice of the final wrapper. Firstly, the occurrence of filter bands from the higher parts of the frequency spectrum for the LFB and OFB feature set requires an investigation of better feature extractors which has more resolution in the high-frequency parts of the spectrum. Modern DL-based feature extraction techniques can be an interesting direction of approach for the given datasets as they would lower the cost of developing handcrafted features. Further, devising a better search optimization strategy in the choice of regularization parameters of CO is an important scope of future work for both model performance and FR of CO. Exploring different percentages of explained variance ratios can be useful in the exploration of PCA for noisy datasets. More DL-based strategies can be investigated in the development of FR methods on larger datasets. Finally, due to the problem of overfitting observed at the final wrapper, a better strategy to fine-tune the hyperparameters can be an important scope of future work on larger datasets with metrics that are sensitive to imbalance from a particular class of machine state for classification problems and metrics that reflect late and early predictions of a machine state differently for regression problems.

List of Acronyms

AI Artificial Intelligence

CF Crest Factor

CO CancelOut

FFT Fast Fourier Transform

FR Feature Ranking

FS Feature Selection

IMMS Institut für Mikroelektronik - und Mechatronik-Systeme gemeinnützige GmbH

LASSO *Linear Absolute Shrinkage and Selection Operator*

MLP Multi-layer Perceptron

ReLU *Rectified Linear Unit*

RMSE Root Mean-Squared Error

MSE Mean Squared Error

RUL Remaining Useful Life

PCA Principal Component Analysis

PdM Predictive Maintenance

RFE Recurcive Feature Elimination

ML Machine Learning

SFS Sequential Forward Selection

SBS Sequential Backward Selection

SF Shape Factor

SOTA state-of-the-art

STFT Short-Time Fourier Transform

List of Figures

List of Tables

Bibliography

[1] Zhang, W., Yang, D. and Wang, H., 2019. Data-driven methods for predictive maintenance of industrial equipment: A survey. *IEEE Systems Journal*, 13(3), pp.2213-2227.

[2] Sammut, C. and Webb, G.I., 2017. *Encyclopedia of machine learning and data mining*. Springer Publishing Company, Incorporated.

[3] Er, M.J., Venkatesan, R. and Wang, N., 2016, October. An online universal classifier for binary, multi-class and multi-label classification. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (pp. 003701-003706). IEEE.

[4] Zheng, F., Zhang, G. and Song, Z., 2001. Comparison of different implementations of MFCC. *Journal of Computer science and Technology,* 16(6), pp.582-589.

[5] Couvreur, C. and Couvreur, I.C., 1998. Implementation of a one-third-octave filter bank in MATLAB.

[6] Botchkarev, A., 2018. Performance metrics (error measures) in machine learning regression, forecasting and prognostics: Properties and typology. *arXiv preprint arXiv:1809.03006*.

[7] Bishop, C.M. and Nasrabadi, N.M., 2006. *Pattern recognition and machine learning (Vol. 4, No. 4, p. 738)*. New York: springer.

[8] Brownlee, J., 2019. A gentle introduction to the rectified linear unit (ReLU). *Machine learning mastery, 6*.

[9] Marcano-Cedeño, A., Quintanilla-Domínguez, J., Cortina-Januchs, M.G. and Andina, D., 2010, November. Feature selection using sequential forward selection and classification applying artificial metaplasticity neural network. In *IECON 2010-36th annual conference on IEEE industrial electronics society* (pp. 2845-2850). IEEE.

[10] Huang, F., Sava, A., Adjallah, K.H. and Wang, Z., 2018, August. Bearings degradation monitoring indicator based on segmented hotelling t square and piecewise linear representation. In *2018 IEEE International Conference on Mechatronics and Automation (ICMA)* (pp. 1389-1394). IEEE.

[11] Kim, T.K., 2017. Understanding one-way ANOVA using conceptual figures. *Korean journal of anesthesiology, 70(1),* p.22.

[12] Duch, W., 2006. Filter methods. In *Feature Extraction* (pp. 89-117). Springer, Berlin, Heidelberg.

[13] Amruthnath, N. and Gupta, T., 2018, April. A research study on unsupervised machine learning algorithms for early fault detection in predictive maintenance. In *2018 5th International Conference on Industrial Engineering and Applications (ICIEA)* (pp. 355-361). IEEE.

[14] Liu, H. and Setiono, R., 1998. Incremental feature selection. *Applied Intelligence, 9(3),* pp.217-230.

[15] Kuhn, M. and Johnson, K., 2013. *Applied predictive modeling* (Vol. 26, p. 13). New York: Springer.

[16] Borisov, V., Haug, J. and Kasneci, G., 2019, September. Cancelout: A layer for feature selection in deep neural networks. In *International Conference on Artificial Neural Networks* (pp. 72-83). Springer, Cham.

[17] Duch, W., Wieczorek, T., Biesiada, J. and Blachnik, M., *2004, July. Comparison of feature ranking methods based on information entropy. In 2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541)*(Vol. 2, pp. 1415-1419). IEEE.

[18] Duan, K.B., Rajapakse, J.C., Wang, H. and Azuaje, F., 2005. Multiple SVM-RFE for gene selection in cancer classification with expression data. *IEEE transactions on nanobioscience,* 4(3), pp.228-234.

[19] Chandrashekar, G. and Sahin, F., 2014. *A survey on feature selection methods. Computers & Electrical Engineering,* 40(1), pp.16-28.

[20] Thakker, H.T., Dave, V., Vakharia, V. and Singh, S., 2020, May. Fault diagnosis of ball bearing using Hilbert Huang transform and LASSO feature ranking technique. In *IOP Conference Series: Materials Science and Engineering* (Vol. 841, No. 1, p. 012006). IOP Publishing.

[21] Freedman, D.A., 2009. *Statistical models: theory and practice.* cambridge university press.

[22] Dayton, C.M., 1992. Logistic regression analysis. *Stat*, pp.474-574.

[23] Karlik, B. and Olgac, A.V., 2011. Performance analysis of various activation functions in generalized MLP architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems,* 1(4), pp.111-122.

[24] Tao, S., Boley, D. and Zhang, S., 2016. Local linear convergence of ISTA and FISTA on the LASSO problem. *SIAM Journal on Optimization,* 26(1), pp.313-336.

[25] Zou, H. and Hastie, T., 2005. Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology),* 67(2), pp.301-320.

[26] Tibshirani, R., 2011. Regression shrinkage and selection via the lasso: a retrospective. *Journal of the Royal Statistical Society: Series B (Statistical Methodology), 73(3),* pp.273-282.

[27] Borgerding, M., Schniter, P. and Rangan, S., 2017. AMP-inspired deep networks for sparse linear inverse problems. *IEEE Transactions on Signal Processing, 65(16),* pp.4293-4308.

[28] Alt, H.W., 2016. Linear functional analysis. *An Application-oriented Introduction.*

[29] Jović, A., Brkić, K. and Bogunović, N., 2015, May. A review of feature selection methods with applications. In *2015 38th international convention on information and communication technology, electronics and microelectronics (MIPRO)* (pp. 1200-1205). Ieee.

[30] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research, 15(1),* pp.1929-1958.

[31] Li, Y., Chen, C.Y. and Wasserman, W.W., 2016. Deep feature selection: theory and application to identify enhancers and promoters. *Journal of Computational Biology, 23(5),* pp.322-336.

[32] POLIKAR, R., The Wavelet Tutorial Second Edition Part I.

[33] Nectoux, Patrick, Rafael Gouriveau, Kamal Medjaher, Emmanuel Ramasso, Brigitte Chebel-Morello, Noureddine Zerhouni, and Christophe Varnier. "PRONOSTIA: An experimental platform for bearings accelerated degradation tests." *In IEEE International Conference on Prognostics and Health Management, PHM'12., pp. 1-8. IEEE Catalog Number: CPF12PHM-CDR, 2012.*

[34] W. F. McGee and G. Zhang, "Logarithmic filter banks," *IEEE International Symposium on Circuits and Systems,* 1990, pp. 661-664 vol.1, doi: 10.1109/ISCAS.1990.112164.

[35] Yamada, Y., Lindenbaum, O., Negahban, S. and Kluger, Y., 2020, November. Feature selection using stochastic gates. In I*nternational Conference on Machine Learning (pp. 10648-10659)*. *PMLR*.

[36] Krishnan, S., Xiao, Y. and Saurous, R.A., 2017. Neumann optimizer: A practical optimization algorithm for deep neural networks. *arXiv preprint arXiv:1712.03298*.

[37] Kingma, D.P. and Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint *arXiv:1412.6980*.

[38] Ruder, S., 2016. An overview of gradient descent optimization algorithms. arXiv preprint *arXiv:1609.04747*.

[39] Beck, A. and Teboulle, M., 2009. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences,* 2(1), pp.183-202.

[40] Hoerl, A.E. and Kennard, R.W., 1981. Ridge regression—1980: Advances, algorithms, and applications. *American Journal of Mathematical and Management Sciences,* 1(1), pp.5-83.

[41] Tax, D.M. and Duin, R.P., 2002, August. Using two-class classifiers for multiclass classification. *In Object recognition supported by user interaction for service robots* (Vol. 2, pp. 124-127). IEEE.

[42] Winkler, D.A. and Le, T.C., 2017. Performance of deep and shallow neural networks, the universal approximation theorem, activity cliffs, and QSAR. *Molecular informatics*, 36(1-2), p.1600118.

[43] Mosallam, A., Medjaher, K. and Zerhouni, N., 2014. Time series trending for condition assessment and prognostics. *Journal of manufacturing technology management*.

[44] Zhao, Z., Li, T., Wu, J., Sun, C., Wang, S., Yan, R. and Chen, X., 2020. Deep learning algorithms for rotating machinery intelligent diagnosis: An open source benchmark study. *ISA transactions*, 107, pp.224-255.

[45] Qazi, N. and Raza, K., 2012, March. Effect of feature selection, SMOTE and under sampling on class imbalance classification. In *2012 UKSim 14th International Conference on Computer Modelling and Simulation* (pp. 145-150). IEEE.

[46] Chawla, N.V., Bowyer, K.W., Hall, L.O. and Kegelmeyer, W.P., 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research,* 16, pp.321-357.

[47] Ha, J. and Lee, J.S., 2016, January. A new under-sampling method using genetic algorithm for imbalanced data classification. *In Proceedings of the 10th International Conference on Ubiquitous Information Management and Communication* (pp. 1-6).

[48] Parsa, M., Mitchell, J.P., Schuman, C.D., Patton, R.M., Potok, T.E. and Roy, K., 2020. Bayesian multi-objective hyperparameter optimization for accurate, fast, and efficient neural network accelerator design. *Frontiers in neuroscience,* 14, p.667.

[49] Alibrahim, H. and Ludwig, S.A., 2021, June. Hyperparameter optimization: Comparing genetic algorithm against grid search and bayesian optimization. In *2021 IEEE Congress on Evolutionary Computation (CEC) (pp. 1551-1559)*. IEEE.

[50] Nishino, T., Takeuchi, S., Saito, T. and Watanabe, I., 2017, June. Fault sound simulations from normal sounds for data-driven prognosis based on human expert and vibration knowledge. In *2017 IEEE International Conference on Prognostics and Health Management (ICPHM) (pp. 266-272)*. IEEE.

[51] Liu, Y., Zhou, Y., Wen, S. and Tang, C., 2014. A strategy on selecting performance metrics for classifier evaluation. *International Journal of Mobile Computing and Multimedia Communications (IJMCMC),* 6(4), pp.20-35.

[52] Farahat, A. and Gupta, C., 2020, November. Similarity-based Feature Extraction from Vibration Data for Prognostics. *In Annual Conference of the PHM Society* (Vol. 12, No. 1, pp. 10-10).

[53] Nezhad, M.Z., Zhu, D., Li, X., Yang, K. and Levy, P., 2016, December. SAFS: A deep feature selection approach for precision medicine. In *2016 IEEE international conference on bioinformatics and biomedicine (BIBM)* (pp. 501-506). IEEE.

[54] Tyagi, V. and Wellekens, C., 2005, March. On desensitizing the Mel-Cepstrum to spurious spectral components for Robust Speech Recognition. In *Proceedings.(ICASSP'05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005. (Vol. 1, pp. I-529)*. IEEE.

[55] Chen, Q., Meng, Z., Liu, X., Jin, Q. and Su, R., 2018. Decision variants for the automatic determination of optimal feature subset in RF-RFE. *Genes,* 9(6), p.301.

[56] Brill, F., Erukhimov, V., Giduthuru, R. and Ramm, S., 2020. *OpenVX Programming Guide.* Academic Press.

[57] Isermann, R., 2005. *Fault-diagnosis systems: an introduction from fault detection to fault tolerance.* Springer Science & Business Media.

[58] Van Dusen, C., 2018. Methods to Prevent Overfitting and Solve Ill-Posed Problems in Statistics: Ridge Regression and LASSO, undated. *As of April, 29*.

[59] Raychaudhuri, S., 2008, December. Introduction to monte carlo simulation. *In 2008 Winter simulation conference (pp. 91-100)*. IEEE.

[60] Ci, B. and Rule, R.O., 1987. Confidence intervals. *Lancet, 1(8531),* pp.494-7.

Appendix A

FR indices

A.1 FAN dataset

The top five features are elaborated in Table A.1 for all the FR algorithms for the FAN dataset. The features are notated in the following manner: {sensor_number}{LFB_band}

Rank	PCA	CO	RFE
1	2_3	3_2	2_5
2	1_4	3_3	2_2
3	3_5	3_1	2_1
4	1_2	3_4	1_1
5	1_3	1_5	1_2

Table A.1 – Top five features for LFB feature set

A.2 PRONOSTIA dataset

The top five broadband features are elaborated in Table A.2 for all the FR algorithms. The features are notated in the following manner: {sensor_number}{statistical feature_index}. Further the statistical features are notated by indexing the first element as 1 and are defined in the given order: {RMS (1), SF (2), Variance (3), Skewness (4), Kurtosis (5), CF (6)}

Rank	PCA	CO	RFE	LASSO
1	2_1	1_3	1_2	1_3
2	2_5	2_1	2_2	1_6
3	1_5	2_3	1_1	1_5
4	2_2	1_1	1_3	2_5
5	2_3	1_4	2_1	N/A

Table A.2 – Top five features for Broadband feature set (PRONOSTIA)

The top five DWT features are elaborated in Table A.3 for all the FR algorithms. The features are notated in the following manner: {sensor_number}{DWT_band}{statistical feature_index}. Further the statistical features are notated by indexing the first element as 1 same as the broadband statistics. The bands of the DWT are defined in the given order: {Band_1, Band_2, Band_3, Band_4}

Rank	PCA	CO	RFE	LASSO
1	2_1_1	1_3_6	1_2_2	1_2_5
2	1_3_2	1_3_5	1_1_2	1_1_5
3	1_3_6	1_1_1	1_3_1	1_1_3
4	1_1_1	1_2_5	1_4_1	1_3_5
5	1_4_3	1_1_6	1_4_3	2_3_5

Table A.3 – Top five features for DWT feature set

The top five features for the LFB feature set are elaborated in Table A.4 for all the FR algorithms. The features are notated in the following manner: {sensor_number}{LFB_band}

Rank	PCA	CO	RFE	LASSO
1	1_8	1_4	2_9	1_10
2	1_1	2_1	2_8	1_3
3	2_9	2_7	2_3	1_4
4	2_2	2_4	1_6	1_7
5	2_8	1_1	1_8	N/A

Table A.4 – Top five features for LFB feature set

The top five features for the OFB feature set are elaborated in Table A.5 for all the FR algorithms. The features are notated in the following manner: {sensor_number}{OFB_band}

Rank	PCA	CO	RFE	LASSO
1	1_8	2_1	2_7	1_9
2	2_4	2_2	1_9	1_3
3	2_5	1_2	1_10	1_4
4	2_3	1_3	1_7	1_7
5	1_5	1_1	1_6	2_10

Table A.5 – Top five features for OFB feature set

Appendix B

Metrics

Parameter	Formulae
MSE	$\frac{1}{N}\sum_{i=1}^{N}\left(y_T - \hat{y}_p\right)^2$
RMSE	$\sqrt{\frac{\sum_{i=1}^{N}\left(y_T - \hat{y}_p\right)^2}{N}}$
MAE	$\frac{\sum_{i=1}^{N}\left\|y_T - \hat{y}_p\right\|}{N}$

Table B.1 – Regression metrics[6]

The correct selection of performance metrics is an important issue in evaluating a model's performance. Further, a wide array of metrics exists to monitor performance on the validation data and is completely scenario dependent[51]. For regression problems, for a ground truth (y_T) and a predicted value (y_p) the various types of metrics are shown in Table B.1[6]. For evaluating PdM scenarios, the metric must have the same unit as the predicted value for interpretability. Hence, Root Mean Square Error (RMSE) and Mean-Absolute Error (MAE) are directly interpretable options as opposed to Mean-Squared-Error (MSE) as it offers a unit that is not interpretable for PdM scenarios.

For classification problems, the confusion matrix is a tool to explore the metrics for evaluating the performance of a classification model. An example of a binary classification problem is taken to illustrate several metrics. The confusion matrix consists of a grid of predicted and actual values depicting the True Positives (TP), True Negatives (TN), False

Positives (FP) and False Negatives (FN) out of a given set of observations K is depicted in Table B.2 for a binary classification problem.

		Ground truth		
		Positive	Negative	Total
Predicted value	Positive	TP	FN	$TP + FN$
	Negative	FP	TN	$FP + TN$
	Total	$TP + FP$	$FN + TN$	K

Table B.2 – Confusion matrix

Accuracy is the quintessential metric for balanced datasets. However, oftentimes datasets are not balanced, and therefore to improve the reliability of the obtained performance, metrics like F1 scores, precision, and recall are considered for such scenarios. This is because, for an unbalanced dataset, it is important to figure out how well the classifier performs in predciting the minority classes. The details of these metrics are displayed in Table B.3.

Parameter	**Formulae**
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$
Precision	$\frac{TP}{TP+FP}$
Recall	$\frac{TP}{TP+FN}$
F1 score	$\frac{2.Precision.Recall}{Precision+Recall}$

Table B.3 – Classification metrics[51]

Further for multiclass classification problems, the Precision, Recall and F1 score are calculated for each classes with an "one vs one" or a "one vs rest" strategy and are averaged in several ways to achieve an overall score for a given problem[3, 41]. A micro averaging involves an overall score of the metrics. Here, a micro averaged F1 score is same as the accuracy and is therefore sometimes sub-optimal for unbalanced scenarios. However, a macro and weighted average of individual performance scores can take class imbalance into account.

Appendix C

Code fragments

C.0.1 Feature extractor code implementation

A short fragment of the feature extractor python code is provided below. A more detailed assistance on the OFB is provided here. [3]

```
def Feature_extract(data):
  for frames in data:
    #Broadband
    broadband_x = extract_broadband(frames) # computes broadband statistics
    #DWT features
    wavelet_list_x = pywt.wavedec(frames, 'db4', level=number_dwtbands-1) # obtains (number_dwtbands-1) DWT bands
    for index_dwt in range(number_dwtbands):
      dwt_x = extract_broadband(wavelet_list_x[index_dwt]) #computes statistical features of these dwt bands
    #LFB features
    lfb_x =logfbank(frames, samplerate=25600, winlen=0.1, winstep=1, nfilt=10, nfft=2 ** 12, preemph=0) #computes 10 features for axis I data
    #Octave FB
    octaves_x = octavefilter(frames, fs=25600, order=6)
    o) # computes the power of the octave FB band
#Octave FB power
def octavefilter(frames, fs, order=6):
  """
  Filter a signal with octave or fractional octave filter bank.
   x: signal frame
  fs: Sampling frequency
    order: Order of Butterworth filter.
  :returns: OFB power and Frequency array
```

[3] https://github.com/SiggiGue/pyfilterbank

```
  """
  # Generate frequency array
  freq, freq_d, freq_u = _genfreqs(limits, fraction, fs) #generates ↘
      frequencies and  removes outer frequency to prevent filter ↘
      error (fs/2 < freq)
  # Get SOS filter coefficients (3D - matrix with size: ↘
      [freq,order,6])
  sos = _buttersosfilter(freq, freq_d, freq_u, fs, order)
  # Create array with SPL for each frequency band
  spl = np.zeros([len(freq)])
  for idx in range(len(freq)):
    sd = signal.decimate(x, factor[idx])
    y = signal.sosfilt(sos[idx], sd)
      spl[idx] = 20 * np.log10(np.std(y) / 2e-5) #OFB power
    return spl.tolist(), freq

# Extract broadband features
from scipy.stats import skew
from scipy.stats import kurtosis
import numpy as np
def extract_broadband(data):
  rms = np.sqrt(np.mean(np.square(data)))
  var = np.var(data)
  skewness = skew(data)
  kurtosis = kurtosis(data)
  SF = np.sqrt(np.mean(np.square(data))) / np.mean(abs(data))
  CF = max(data) /= np.sqrt(np.mean(np.square(data)))

  return [rms, var, skewness, kurtosis, SF, CF]
```

C.0.2 FR code implementation

This subsection lists some small fragments of code used for the FR algorithms.

PCA

A short python code to develop the PCA implementation.

```
  def PCA(X_train):
    scaler = preprocessing.StandardScaler()
    scaled_X = scaler.fit_transform(X_train)
    pca = PCA(0.95)
    pc = pca.fit_transform(scaled_X)
    evr = pca.explained_variance_ratio_.reshape(-1,1)
     pca_FRweights = np.sum(evr*np.abs(pca.components_),axis=0))
    indexes= np.argsort(abs(pca_FRweights))[::-1]
```

```
9       return indexes
```

RFE

A short python code to develop RFE implementation.

```
1   def RFE(X_train,y_train):
2     for i in range(1,n+1):
3       lr = linear_model.LogisticRegression()
4       rfe = RFE(estimator= lr, step=1, n_features_to_select=i)
5       rfe.fit(X_train, y_train)
6       arr = [x for x,y in enumerate(rfecv.get_support()) if y==True]
```

LASSO

A short python code to develop LASSO hyperparameter search implementation.

```
1 time_to_LASSO_param= []
2 MSE = []
3 alpha_range  = np.linspace(0.01, 1, num=50)
4 for alpha_val in alpha_range:
5   start = time.time()
6   lasso = linear_model.Lasso(alpha=alpha_val,max_iter=20000)
7   lasso.fit(x_train,y_train)
8   selected_feat_indices = feature_index_sort(np.abs(lasso.coef_))
9   end = time.time()
10  y_hat = lasso.predict(x_val)
11  mse_temp = mean_squared_error(y_hat,y_val)
12  MSE.append(-1 *np.sqrt(mse_temp))
13  time_to_LASSO_param.append((end-start))
```

C.0.3 CO

Another short snippet for performing hyperparameter tuning of CO is provided below. A more generic code implementation of CO can be explored in [16][4].

```
1 for l1 in lambda_1_range:
2   for l2 in lambda_2_range:
3   start = time.time()
4   inputs = keras.Input(shape=(X_train.shape[1],))
5   x = CancelOut(activation='softmax', lambda_1=l1, ↘
        lambda_2=l2)(inputs)
6   x = layers.Dense(neuron, activation="relu")(x)
```

[4] https://github.com/unnir/CancelOut

```
outputs = layers.Dense(y_train_encoded.shape[1], ↘
    activation='softmax',kernel_initializer='random_normal')(x)
model = keras.Model(inputs=inputs, outputs=outputs)
 model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=1e-2),
loss='categorical_crossentropy',
metrics=['acc'])
model.fit(x_train, y_train, epochs=200, batch_size=512, ↘
    validation_data=(x_val,y_val))
cancelout_feature_importance_sigmoid = model.get_weights()[0]
end = time.time()
perf=model.evaluate(x_test, y_test)
```

C.0.4 Final wrapper

Finally, a short snippet of the final wrapper with a linear model (Logistic Regression) is displayed below:

```
from sklearn.metrics import f1_score,accuracy_score
feature=[] # for the wrapper to perform over FR indices
for index in range(len(selected_feat_indices)):
  lr = LogisticRegression(C=1, penalty= 'l1', solver= 'liblinear')
  temp = selected_feat_indices[index]  #current FR index
  feature.append(temp)
  reggr = lr.fit(X[:,feature], y)
  y_pred = lr.predict(feat_label_np_test[:,feature])
  acc = accuracy_score(y_test, y_pred)
  f1 = f1_score(y_test, y_pred , average="macro")
```