FAULT PROGNOSTIC USING REPRESENTATION BASED CLASSIFICATION BY DERIVING HEALTH INDEX WITH DOMAIN ADAPTATION

A PROJECT REPORT

Submitted by

SHRIRAM G	2018506116
HARIHARA KRISHNA V	2018506035
AJAI B	2018506010

Under the supervision of

Dr .P.Anandha Kumar, M.E, Ph.D.

In partial fulfilment for the award of the degree

Of

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY

MADRAS INSTITUTE OF TECHNOLOGY CAMPUS

ANNA UNIVERSITY, CHENNAI – 600044

JUNE 2022

MADRAS INSTITUTE OF TECHNOLOGY CAMPUS

ANNA UNIVERSITY

Chennai 600025

BONAFIDE CERTIFICATE

Certified that this project work report

"Fault Prognostic Using Representation Based Classification by Deriving Health Index With Domain Adaptation"

is the Bonafide work of

"SHRIRAM G (2018506116), HARIHARA KRISHNA V (2018506035), AJAI B (2018506010)"

who carried out this project work under my supervision.

Signature	**Signature**
Dr. Dhananjay Kumar	**Dr. P.Anandha Kumar,Ph.D**
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Professor	Professor
Department of Information Technology	Department of Information Technology
MIT Campus, Anna University	MIT Campus, Anna University
Chennai – 600044	Chennai – 600044

ACKNOWLEDGEMENT

It is essential to mention the names of the people, whose guidance and encouragement helped us to complete this project.

We express our thankfulness to our project supervisor, **Dr.P.AnandhaKumar**, Professor, Department of Information Technology, MIT Anna University, for providing invaluable support and assistance with encouragement which aided to complete this project.

We are thankful to the panel members **Dr. Radha Senthilkumar, Dr.B.Lydia Elizabeth and Mrs.P.Jayanti,** Department of Information Technology, MIT Campus for their invaluable feedback in reviews.

Our sincere thanks to **Dr. Dhananjay Kumar**, Head of the Department of Information Technology, Teaching and Non-teaching staffs of Information Technology, MIT Campus, for catering to all our needs and supporting us throughout this project.

We express our gratitude to our respected Dean of MIT Campus, for providing excellent computing facilities to complete this project.

SHRIRAM G	**2018506116**
HARIHARA KRISHNA V	**2018506035**
AJAI B	**2018506010**

ABSTRACT

Deep neural networks (DNNs) have lately demonstrated impressive performance in a variety of image identification tasks. Existing deep neural network models, on the other hand, are computationally and memory intensive, preventing their use in devices with limited memory or in applications with severe throughput requirements. Thus, reducing the number of parameters and making a smaller model that can run under constraints of the edge devices is the key challenge. Most of the existing algorithms face difficulty in localization of feature space. In order to address this issue, an innovative solution that classifies the samples based on their representatives is proposed here. The purpose of this study focuses on building a classification model that has limited or no hyper parameters and with limited trainable parameters. Thus reducing the size and training time of the classification model without losing knowledge of the model. Bearing fault detection is chosen as the base application for validating the proposed work, but it is not limited to it. In comparison to traditional classification models, this statistical model shows improved performance in classification of faulty samples over 10% to 30%. This technique is not only limited to handling binary classification but also can be extended to applications involving more than two target classes.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

ABBREVIATIONS	FULL FORM
CADNN	Compressed and Accelerated Deep Neural Network
RUL	Remaining Useful Life
PDA	Performance Degradation Assessment
RBE	Rolling Bearing Elements
LSTM	Long Short-Term Memory
RNN	Recurrent Neural Network
IMNN	Intensification and Metamorphosing based Neural Network
SWOGUT	Synthetic Weighted Over-Sampling and Guided Under-Sampling Technique
RBC	Representation Based Classification

SYMBOLS	MEANING
$\sum$	Sigma
$>$	Greater than
$<$	Lesser than
$\geq$	Greater than or equal to
$\leq$	Lesser than or equal to

CHAPTER 1
INTRODUCTION

1.1 OVERVIEW

The revolution which technology has brought can be inferred from the quality, quantity and the speed of delivery of the industry 4.0’s products [25]. The amount of automation brought into the existing machines has made day-day life simple and fast. Getting to know the remaining useful life of any resource might be an added advantage for many (Pre-Maintenance (prevent), Cost efficiency) [28, 29]. Prognosis of faulty components [26] in such machinery might solve some dangerous after effects to firms which completely depend on the machinery. Domain adaptation is a technique wherein a source dataset is trained by a particular Neural Network to achieve commendable accuracy and then a target dataset (which should be relevant to the source dataset) is tested to obtain the same. In this technique, it is observed that no step can be taken up without any prior/historical knowledge.

Domain adaptation helps in implementing transfer learning as it transfers source knowledge for the target dataset. The main takeaway in implementing Domain Adaptation is the amount of time and resource that is saved while solving a problem. This solves the problem of recursive training of machine learning models from scratch. Fault prognosis mainly suffers from the nature of input data such as the ratio of healthy data instances is usually huge compared to faulty data samples. Hence it is necessary to balance the input data effectively. Next, the present classification models are not that accurate in classifying fault samples as it has large decision boundaries and data’s are ambiguous. Deep learning techniques which show good performance but still require large resources in terms of computation time and memory. So it is necessary to compress and accelerate the classification model to make it work real-time in edge devices without a considerable amount of latency.

1.2 RESEARCH CHALLENGES

Fault diagnosis usually proceeds with performance degradation assessment and then deriving a health indicator which is then used for RUL prediction. But need of final failure labels to train the appropriate models is inevitable. But with the available CWRU dataset, it was a great challenge to deal with this issue. Developing GAN-based implementation was strenuous because of usage of high RAM and GPU-based resources. Requires target failures samples to train the assessment models. Although only healthy data were required to train the Gaussian mixture model, the results are influenced by the number of mixture components and require to be judged appropriately. It requires a priori knowledge of the failure classes to train the classifier. CNN based implementations are computationally expensive. Prescribed as time series but applied CNN as fundamental. Existing Neural network used for training, introduced Bayesian method. It is necessary to build a domain invariant classifier which can classify the faulty samples with less information about the nature of dataset and able to perform the classification with less time and space complexity.

1.3 OBJECTIVE

This project focuses on potentially avoiding failures from systems using predictive measures obtained from previous occurrences and finding the remaining useful life of the machinery. The Main objective of this study is to use effective sampling methods and classification models which can compete with existing deep learning neural networks but requires limited resources in order to run on small edge devices. Usually accelerometer and encoders are used for getting necessary inputs for training a prognostic model. The accelerometer helps to get vibration signals of hardware components which is then processed to analyse the heath state of it. The encoder is often used to make note of data rotor velocity. In this way, a dataset containing vibration signals from bearing is used to train and validate the proposed fault prognostic model. It is necessary to build a domain invariant classifier which can classify the faulty samples with less information about the nature of dataset and able to perform the classification with less time and space complexity.

1.4 SCOPE OF THE PROJECT

- In today’s world, the amount of automated machinery being used in Industry 4.0 to process, manufacture or even reuse might be humongous quantitatively.
- Qualitative performance improvement should be done, but at the right time.
- Deep learning fusion with transfer learning (prognosis) might yield the right benefits

1.5 CONTRIBUTION

Design of the system is divided into three modules.

- Domain Invariant Classifier
- Representation Based Classification for accelerated classification of data samples.
- Health Index derivation for enhanced fault prognostic.

First one involves training the prognostic model. Second module involves deployment of the trained model. Series of data engineering processes (Fig 4.1) are carried out to process the raw input data into usable information. Sampling of an imbalanced dataset is carried out in the pre-processing stage. The predictive model is designed in a way that training is done with historical logged data. Once the model is trained and deployed into the deployment centre, the real-time streaming data is fed to the trained prognostic model to get the live health state of chosen mechanical components. The designing of classification models is where the proposed algorithm is implemented which can classify the given data instance as a healthy or faulty component.

1.6 ORGANIZATION OF THE REPORT

The remaining portions of the project are depicted as follows:

- Chapter 2 describes the literature survey of the project, which discusses about how the authors previously had contributed to this field of research
- Chapter 3 describes our proposed work which includes Representation based classification, HI & GANDA
- Chapter 4 describes the implementation of the proposed novelty
- Chapter 5 describes evaluation, results of the project, and compares the results with existing algorithms
- Chapter 6 describes the conclusion of the project and the future work that can be carried out to improve the work in this field of research
- Chapter 7 and 8 describes the appendix sections I and II and Finally
- Chapter 9 describes the references section which includes the research papers with year of publication and authors, that was referred to proceed the research

1.7 Exploratory Analysis of Dataset

The design of the Case-Based Fault prognosis dataset [21] is to provide discrete data sets of established good and unhealthy conditions for both bearings and gears. The dataset owned in this project is the Case Western Reserve University Bearing dataset as it is freely distributed with all required attributes to provide effective classifications. Acceleration data were gathered at various points near and distant from the bearing over investigations using a 2 hp Valiant Industrial motor. The precise test settings of the motor, as well as the bearing problem status for each experiment, have been painstakingly documented on these web pages. Table 1.1 gives an overview of the attributes present in the CWRU dataset.

Table 1.1 Overview of attributes in CWRU dataset

Attributes	**Name/Function**
DE_time	Drive end accelerometer data
FE_time	Fan end accelerometer data
BA_time	Base accelerometer data
RPM	RPM during testing
Motor Load	Read Error Rate
Fault diameter	Size of fault seeded
Fault	Type of fault

Seeding motor bearings with defects was done using electro-discharge machining (EDM). At the internal racetrack, rolling element (i.e. ball), and outer speedway, faults with different diameters from 0.007 inches to 0.040 inches were injected sequentially. In the test motor, damaged bearings were repaired, and vibration information was analyzed for engine workloads ranging from 0 to 3 kilowatt (motor speeds of 1797 to 1720 RPM).

The flaws in conventional bearings, single-point drive terminals, and fan ends were all documented. Investigations on drive end bearing were performed out at frequencies of 12,000 and 48,000 sample rate. The output from the fan end components was recorded at a rate of 12,000 sample rate.

The CWRU dataset is a structured dataset but still an unbalanced dataset. Since the dataset consists of more than 120 million rows and including the fact that the data collected for bearing has single normal condition with multiple faulty conditions. Hence the dataset tends to be imbalanced as the number of faulty sampled vibration signals is huge compared to healthy sampled vibration signals. The below figure shows how imbalanced the dataset is wherein red dots represent the faulty data samples and the blue dots represent the healthy data samples. Also, only selected features will contribute towards the classification of bearing faults, hence it is necessary to pick the optimal subset of the features that contributes more towards the prediction.

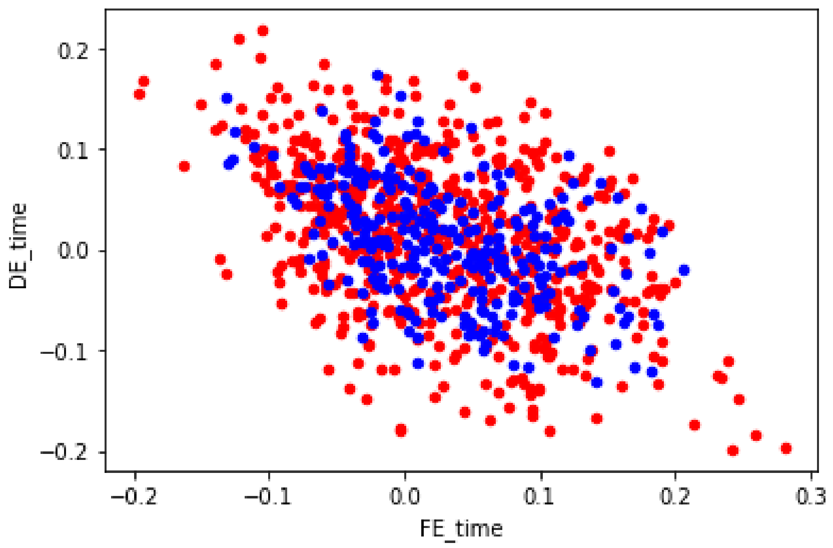


Fig 1.1 scatter plot of the initial dataset

The scatter plot of the initial dataset in Fig 1.1 shows how the healthy and unhealthy data samples are distributed and varies in count.

CHAPTER 2
LITERATURE SURVEY

2.1 DOMAIN ADAPTATION

Yifei Ding et.al [1] has implemented a regression network-based domain adaptation called deep subdomain adaptive regression network (DSARN). The technique consists of a regressor and an extractor, wherein the input is first passed into the extractor to obtain high-dimensional hidden representation and then sent to a regressor which gives a regression prediction by mapping the high dimensional hidden representation with the target data, which happens in any Convolution Network. Dongdong Zhao et.al [2] has proposed a CNN based adversarial domain adaptation. The procedure of this technique is near to absolutely same as the implementation proposed by Yifei Ding et.al.

NacWoo Kim et al. [3] presented a time-series based domain adaptability technique that utilizes a convolutional network as the feature extractor's system and a feed-forward network with one straighten layer, incorporating GRL (gradient reversal layer), and two dense tiers as the domain classifier's network. A time-variant system is one whose output response is contingent on both the moment of observation and the moment of delivery of the input signal. Jialin Li et al. [4] presented an RUL prediction method that uses a deep convolutional neural network (DCNN), Bayesian optimization, and adaptive up sampling (AdaBN).This technique too happens to be similar to the previous works mentioned above. Yongchao Zhang et.al [5] has proposed a technique named DCDAN which follows the same steps as any convolutional network.

Since majority of the existing works have focused on the usage of Convolutional networks and little to nothing focus on implementation of other networks leads to the understanding that redundant research may be diverting the objectives of our project.

2.2 PERFORMANCE DEGRADATION ASSESSMENT

The role of performance deterioration evaluation is to design a health indicator that really can properly monitor bearing deterioration and warn of an unanticipated surge in deterioration intensity. Various signal-processing techniques have been used to retrieve common characteristics from bearing vibration data. [6]After that, the retrieved features are used to create a deterioration indicator for determining bearing health. Understanding the susceptibility of these parameters to the initiation and maintenance of bearing flaws is one of the hurdles in the detection stage. Furthermore, the breakdown data collected by the subspace employed to frame the health index should be more specific.

Numerous publications covering these concerns and laying out various methodologies for evaluating bearing quality deterioration have indeed been developed recently. Using a blend of wavelet transform and self-organizing maps, Qiu et al. [7] established a deterioration index for evaluating bearing effectiveness deterioration. The feature vector used to construct the deterioration index was created using the root mean square (RMS), sample variance, and crest factor of the bearing non - stationary signals and related envelope signals. Pan et al. [8] emphasized the importance of selecting appropriate indices for efficient performance deterioration, and so a complementary index called as spectral entropy for REB performance testing was proposed. Pan et al.[9] used lifting wavelet packet decomposition with fuzzy c-means to construct a health index for tracking bearing deterioration. The strengths of wavelet packet terminals were utilized as a feature space for the development of degradation indices. The techniques proposed in the journals, on the other hand, required final failure examples in order to train the simulation approaches.

Relying on locality preserving projections and Gaussian mixture models, Yu [10] proposed a new process for assessing the deterioration index in bearings. Despite the fact that only healthy data was required to train the Gaussian mixture model, the quantity of mixture components has an influence on the outcomes, which must be judged appropriately. Ali et al. [11] used time-domain data and empirical mode decomposition (EMD) energy entropy as inputs to an artificial neural network to develop a health score for assessing bearing performance deterioration. However, the strategy requires foreknowledge of the failure types in terms of training the classifier.

2.3 SUMMARY

- Requires target failures samples to train the assessment models
- Although only healthy data were required to train the Gaussian mixture model, the results are influenced by the number of mixture components and require to be judged appropriately
- It requires a priori knowledge of the failure classes to train the classifier.
- CNN based implementations are computationally expensive
- Prescribed as time series but applied CNN as fundamental.
- Existing Neural network used for training, introduced Bayesian method

It is necessary to build a domain invariant classifier which can classify the faulty samples with less information about the nature of dataset and able to perform the classification with less time and space complexity.

CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN

3.1 INTRODUCTION

The solution to the problem statement is given as three separate modules.

1. Domain Invariant RUL predictor.
2. Representation based classification
3. Metamorphosing of features into Health degradation indicator.

Thus, integrating these three modules, a complete Enhanced Domain Invariant RUL predicted using Compressed and accelerated Deep Neural Network by deriving Health degradation indicator can be obtained.

3.2 SYSTEM DESIGN

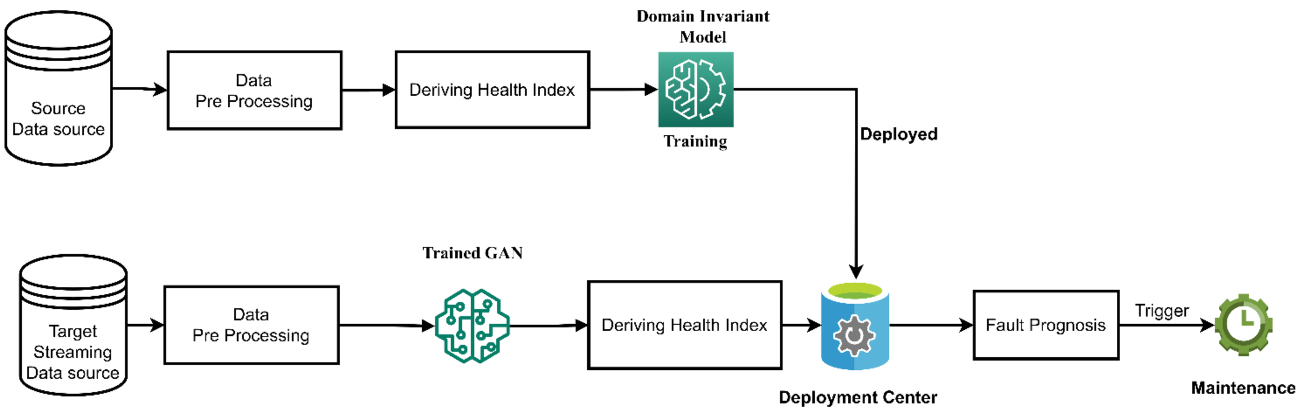


Fig 3.1 - Overall design of system

Fig 3.1 shows the design of the system. Design of the system is divided into two phases. First one involves training the prognostic model. Second module involves deployment of the trained model. Series of data engineering processes (Fig 4.1) are carried out to process the raw input data into usable information. Sampling of an imbalanced dataset is carried out in the pre-processing stage. The predictive model is designed in a way that training is done with historical logged data. Once the model is trained and deployed into the deployment centre, the real-time streaming data is fed to the trained prognostic model to get the live health state of chosen mechanical components. The designing of classification models is where the proposed algorithm is implemented which can classify the given data instance as a healthy or faulty component.

3.3 BLOCK DIAGRAM

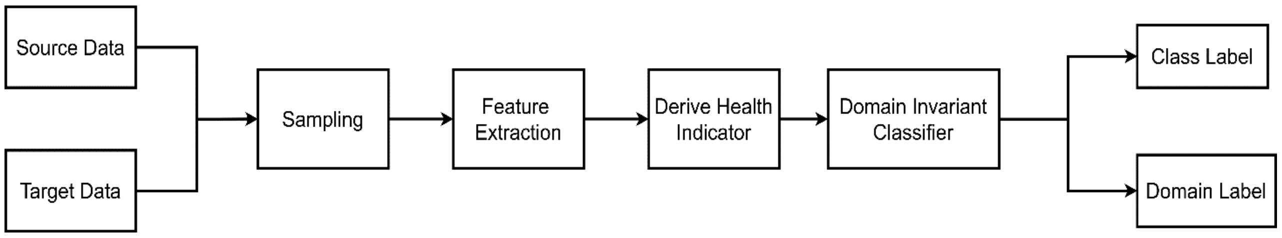


Fig 3.2 - Block diagram of proposed work

The block diagram (Fig 3.2) depicts the flow of algorithm from sampling to predicting the RUL. It starts with sampling of imbalanced dataset and followed by feature extraction. The extracted features are then converted into Health degradation indicator before fed into Domain Invariant Classifier. Finally predicted RUL will be the output from Domain Invariant Classifier.

3.4 SYSTEM ARCHITECTURE:

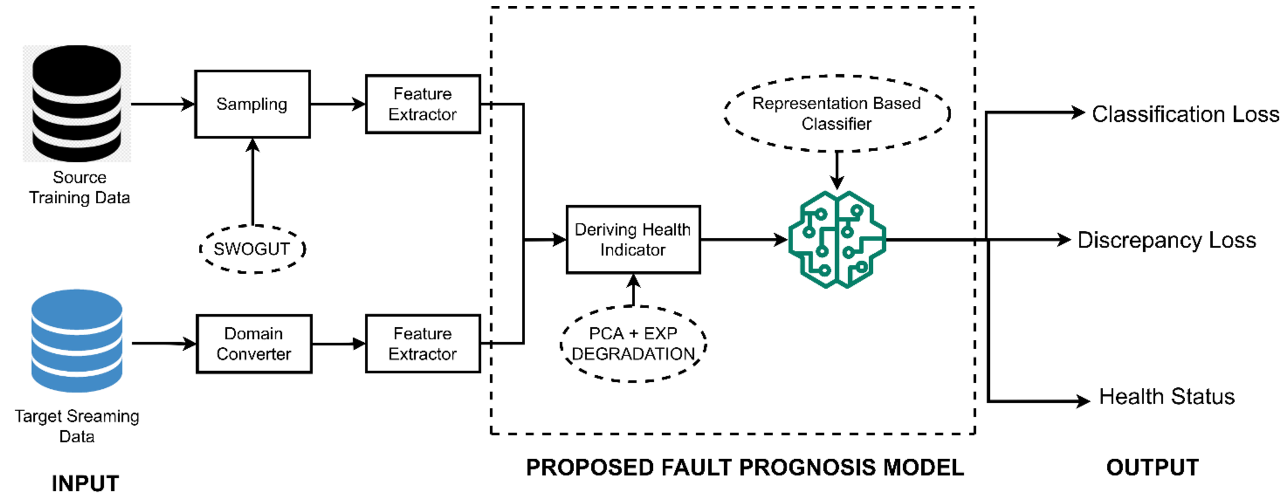


Fig 3.3 - Architecture diagram of proposed work

The Architecture diagram (Fig 3.3) show how different components of proposed work are structured shows the architecture of the designed classification model. The flow starts with pre-processing of the dataset. For pre-processing the procedure followed was certain data engineering processes such as data munging and data standardization are carried out. As CWRU dataset which has a huge difference between ratio of healthy to faulty data instances. The dataset is unbalanced, as the source of this dataset contains very few records for normal conditions. Thus, to avoid over fitting of the classification model, it is important to balance the dataset before forwarding it to the classification model.

For balancing the dataset a novel sampling technique named Synthetic Weighted Over-sampling and Guided Under-sampling Technique (SWOGUT) was used here. The main advantage of using SWOGUT is that, this sampling technique follows a heuristic approach to under-sampling and oversampling of difference class data instances rather than randomly removing or adding data of the majority or minority class. Also, SWOGUT is a hybrid sampling technique as it can be used to over-sampling as well as under-sampling of data samples at the same time. Once the dataset is balanced it is then forwarded to a classification model for training and to predict the health state of the chosen mechanical components. The proposed classification model RCB is again divided into two sections. In the first phase the model is trained by fixing the initial representative followed by the growth and fixing of representatives. After training, the model is ready to predict the health state of the component with all the information about the representatives of the given data instances. The detailed implementation and working of the model on how it predicts the output class outperforming the other existing models is discussed in chapter 4.

3.5 HEALTH INDICATOR ARCHITECTURE

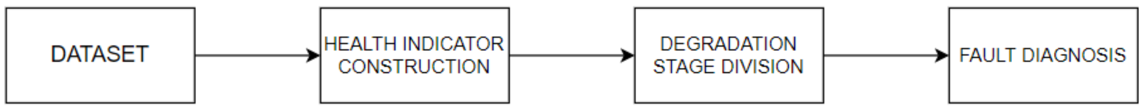


Fig 3.4 Workflow in HI construction

- The common metrics of prognosis are Monotonicity, Prognosability, trend ability, robustness, correlation etc. Here Monotonicity metric is used to well adopt the feature selection process.

$$Monotonicity(x) = \frac{1}{m}\sum_{j=1}^{m}\frac{|\text{number of positive diff}(x_j) - \text{number of negative diff}(x_j)|}{n-1}$$

…… (3.1)

Where diff(x_j) is the lag difference of specific feature.

- The correlation index reflects the degree of correlation and lies between zero and one. The larger the correlation index, the greater the time correlation between the feature and equipment performance degradation; in other words, the degradation feature can better describe the equipment degradation process.
- The term "prognosability" refers to the ability to distinguish between healthy and faulty equipment, as well as the dispersion of the traits. The prognosis values range from zero to one. The closer the prognostic value is to one, the better the prognosis.
- Trendability reflects linear correlation of a feature F and the range is zero to one as well.
- Robustness reflects the degree of fluctuation of the feature vector and its value lies between zero and one.

3.6 SUMMARY:

- Novel Domain Invariant RUL predictor is proposed by using Representation Based Classification.
- The performance of proposed Domain invariant RUL predictor is improved by metamorphosing the feature into health degradation indicator.
- New Representation based classification technique is proposed to speed up the classification problem without compromising the accuracy of classification.

The results from various metrics shows that the proposed Representation based classification by deriving heath indicator of data samples has improved classification performance by 10% to 30% then traditional classification models. Also from the results obtained during classification of target dataset, it is evident that the proposed model is capable of classifying the data samples domain invariantly.

CHAPTER 4
ALGORITHM DEVELOPMENT AND IMPLEMENTATION

4.1 REPRESENTATION BASED CLASSIFICATION:

The dataset obtained from CWRU is not in a balanced condition as the ratio of healthy to faulty bearing is uneven. Most of the available sampling techniques are of three types, they are: oversampling, under-sampling, and hybrid sampling [22] which includes both oversampling and under sampling. A new advanced sampling technique is required to balance the CWRU dataset as need of evenly distributed data instances without losing much information is inevitable. The advanced sampling technique called SWOGUT – synthetic weighted oversampling and guided under sampling technique is used in our work. This sampling work is performed with the help of python libraries in the google collab environment.

The flow of the pre-processing steps is depicted in Fig 4.1 which represents the flowchart of the preprocessing phase. A python script is built to automate the process of organizing, formatting and balancing the dataset based on the conditions of chosen bearing. The script implements the following processes:

1. Loading dataset (in .mat format)
2. Converting the .mat files to csv files.
3. Performs filtering of data by removing single value columns
4. Data munging process is then carried out to remove unwanted features which may affect the performance of the classification model.
5. As the features from the dataset are of different scales, it is necessary to bring all the features into a common scale to avoid overfitting of the model.
6. For removing the least correlated features, the Recursive Feature Elimination method is used.

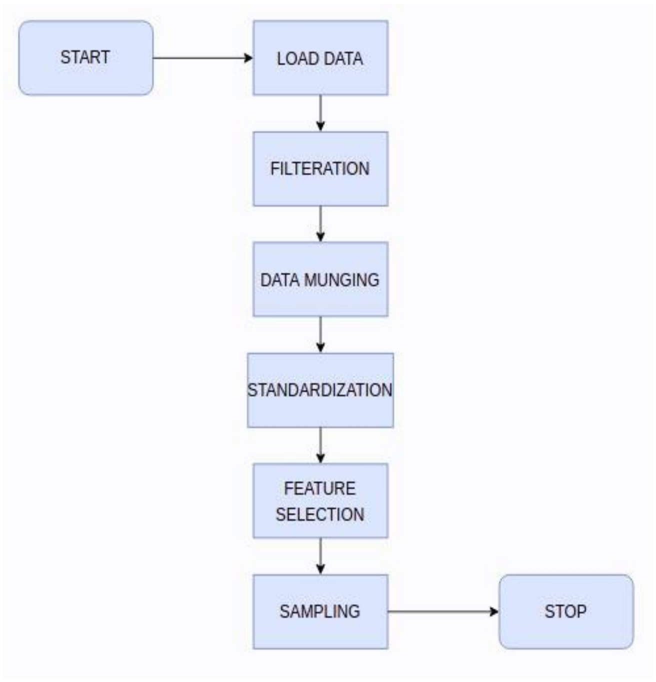


Fig 4.1 Flowchart for preprocessing

To bring about standardization across all the features, all attributes from the CWRU dataset were normalized by removing the average. This step was mandatory to get all the feature parameters into the same scale before applying any machine learning model. This is one of the critical steps in pre-processing because if some features have extremely high values, then the classification of the model will shift towards it.

There are a lot of methods available for selecting appropriate features out of which Scikit-Learn's RFECV function was picked to select features that are more appropriate in classifying the output class variables than any other methods. Here feature elimination is used to select only the features which are highly correlated to the target variable and to eliminate the features which are least or not at all contribute to the classification of data samples. By this way time complexity is reduced for model fitting and to avoid unnecessary shifting of weights of model during training.

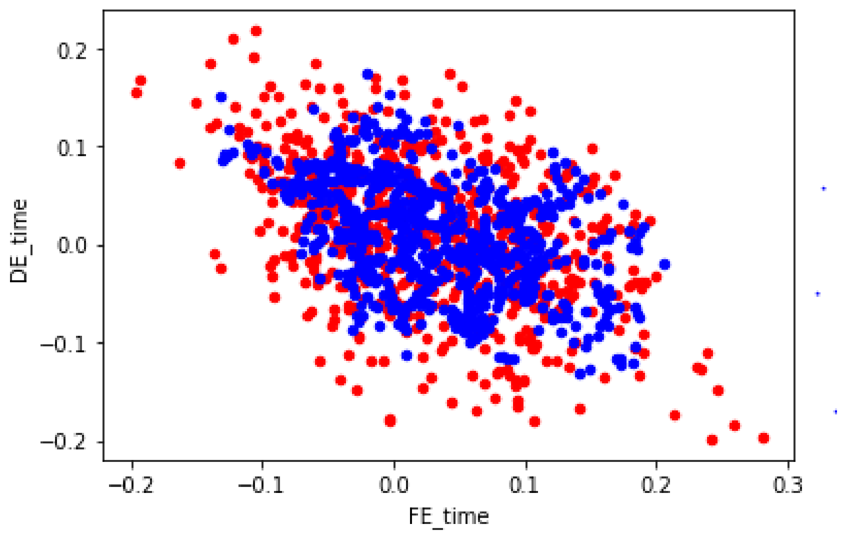


Fig 4.2 CWRU dataset sampled with SWOGUT.

From Fig 4.2 it's obvious that SWOGUT causes the learner to experience different kind of experience of under-sampling, with the minority class getting a larger presence in the training set at advanced degrees of under-sampling.

A reasonable combinational approach to perform over-sampling including under-sampling has been carried out using the advanced sampling technique SWOGUT, in which the ratio of the minority class data samples is boosted by synthesizing the artificial data samples. Similarly, the ratio of majority class samples is under-sampled in a guided fashion rather than deleting randomly picked samples. The inspiration gathered to develop this approach is fed by a method that proved successful in recognizing handwritten characters (Ha & Bunke, 1997).

Algorithm 1 : RBC(Co, data, c_label) → N, C(n)

```
/*      This function returns the representatives of input dataset instances along with their coherence value and associated data members.      */
/*      Co is the minimum coherence value that is expected for every representatives      */
/*      data is the preprocessed input data samples.      */
/*      c_label contains one hot encoding of the target variable corresponding to input data samples.      */
/*      N is the number of fixed representatives for the given dataset.      */
/*      C(n) is the coherence value of the Nth representative      */
begin
        mean_dist -> mean_distance(calculate_rep(data),data)
        reps <- []
        cluster_coherence <- []
        cluster_members <- []
        while True:
                prev_centriod_len -> len(reps)
                for i <- 0 to i <- data.length:
                  nearest_rep  -> nearest data representative to data[i]
                  if min_dist > mean_dist do:
                   create a new representative similar to data[i]
                  else do:
                   update the nearest representative's attributes
                for instance<- 0 to instance<- reps.length do :
                  if cluster_coherence[instance] <= Co and (1 - cluster_coherence[instance]) <= Co do:
```

```
                split the cluster corresponding to the target variable and update
            coherence values to 1.
             update(current_reps_len)
             **if current_reps_len == prev_centriod_len do :**
               **break**
     **return (reps, cluster_coherence, cluster_members)**
**end**
```

Algorithm 2 : CLASSIFY(reps, c_coherence, c_member, K, data) → target_label

---

```
/*     This function does the classification by finding the relevant representative
of the given data sample and by processing the representative's attribute values.
/*     reps is the resultant representatives of the given data instances          */
/*     c_coherence is the value calculated in the process of finding the
representatives.                                                                   */
/* c_member groups the representatives with their associated data instances       */
/*  K decides the number of representatives need to be considered for classifying
the given data sample                                                              */
begin
      neighbour_reps <-  get_neighbors(reps, cluster_coherence, instance,K)
      sum_of_distance <-sum of distance between data and neighbours
      sum_of_coherence <-  sum of coherence value of neighbour
     representatives
      sum_of_cluster_members <- total number of data instances grouped by
     K representatives.
      neighbours_score <-  []
```

```
  for i  <- neighbour_reps:
    neighbours_score_temp <-  []
    neighbours_score_temp.append(i.rep)
    neighbours_score_temp.append(i.coherence_value /
sum_of_coherence)
    neighbours_score_temp.append(1 - dist(i,data) / sum_of_distance)
    neighbours_score_temp.append(i.cluster_members.length) /
sum_of_cluster_members)
    neighbours_score.append(neighbours_score_temp)
  final_neighbour_score <-  []
  **foreach** i  <- neighbours_score do :
    final_neighbour_score.append(i.coherence_value, i.dist_value,
i.cluster_member)
  normalized_final_neighbour_score <-  []
  **foreach** i <- final_neighbour_score:
    normalized_final_neighbour_score.append(i/sum(final_neighbour_sco
re))
  predicted_class <-  0
  for i <- 0 to i<- neighbour_reps.length :
    predicted_class += neighbour_reps.coherence_value *
normalized_final_neighbour_score[i]
  return predicted_class
**end**
```

A Representation based classification using adaptive self-organization technique is proposed as a prognostic classification model. This approach is inspired by following three existing algorithms such as Adaptive resonance theory, unsupervised learning (clustering techniques) and self-organizing map. This algorithm works in two phases. First phase is the training phase where the centroid is fixed and with the centroid as a reference representation is created by adaptive and self-organization of nodes (Representatives) based on proposed criteria. The model is iteratively trained by growing, fixing and learning the representation of the given dataset. One of the main advantages of the learning phase is that it can expand and reduce the feature space by adjusting the expected coherence value that each representative should have.

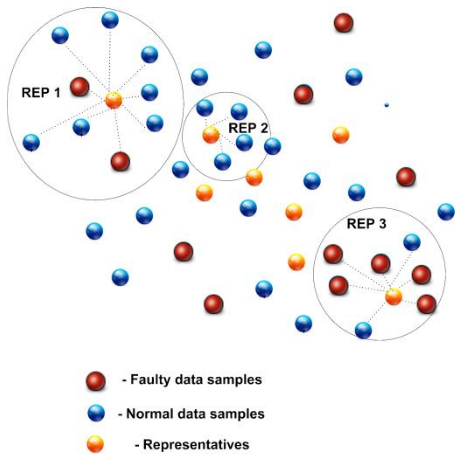


Fig 4.3 Learning phase of RBC- Architecture

Fig 4.3 explains how the learning phase of the Representative Based Classification model works in classifying the data samples. Here is where the actual training that is fixing the representative is done. First the centroid is fixed as an initial representative. The growth of the number of representatives is controlled by two factors. They are the mean distance (1) between the centroid vs all the data samples and the expected coherence value of each of the representatives. If either of the constraints is violated then the representative will be splitted into two with one set of data samples associated with one representative and another set of data samples associated with another representative with coherence value of 1 in both representatives.

The first constraint is to ensure that the radius covered by each representative should not exceed the threshold value. Here the mean distance between centroid and all the data samples is taken as the threshold. And this can be tuned to the nature of the dataset used. Second constraint is the expected coherence value that each representative should possess. The coherence value of each representative is calculated using formula (2). The expected coherence value is a hyper parameter whose value is fixed by the administrator depending on the density of data samples and resources available such as time and memory. Because here each representative is considered as a trainable parameter.

So as the expected coherence value increases, the number of representatives with such coherence value also increases resulting in increased time and memory complexity.

$$dist(data\ ,\ representative) > \frac{\sum_i dist(centroid, data_i)}{N}\ , i = 1, 2 \ldots N \qquad \ldots\ (4.1)$$

$$coherence(reresentative) = \frac{\sum_i class1(representative.member_i) + \sum_i class2(representative.member_i)}{\sum_i class1(representative.member_i)}\ , i = 1, 2 \ldots N \qquad \ldots\ (4.2)$$

The growth of the number of representatives is controlled by the below formula (4.3)

$$coherence(reresentative) < expected\ coherence\ and\ [1 - coherence(reresentative)] < expected\ coherence \qquad \ldots\ (4.3)$$

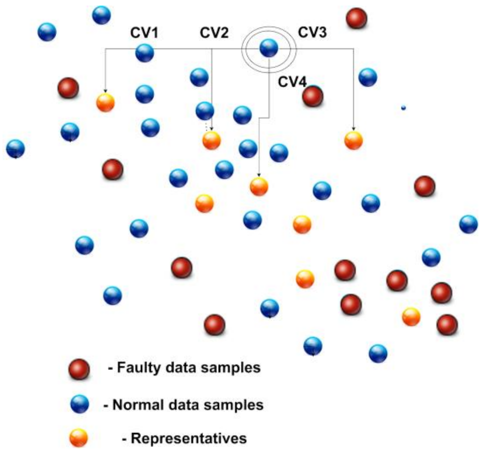


Fig 4.4 Classification phase of RBC- Architecture

Second phase of the algorithm involves classification of data samples using learned representatives of the given dataset. Once the representatives are fixed in their position with coherence value, it is apparent to classify the unseen samples using the information from memory of the representatives. There are a series of steps to be followed to normalize the coherence value of each representative and to generate the score for each representative.

$$neighbour.n\_dist = \frac{neighbour.dist}{\sum_i neighbour_i.dist} \quad ,i = 1,2 \ldots K \qquad \ldots (4.4)$$

$$neighbour.n\_coherence = \frac{neighbour.coherence}{\sum_i neighbour_i.coherence} \quad ,i = 1,2 \ldots K \qquad \ldots (4.5)$$

$$neighbour.n\_density = \frac{neighbour.density}{\sum_i neighbour_i.density} \quad ,i = 1,2 \ldots K \qquad \ldots (4.6)$$

Score for individual attributes of the representative is calculated for every K nearest representative using formulas (4), (5) and (6). The normalized final score of each representative is calculated by combining all the above formula (7).

$$\sum_i neighbour_i.score = \frac{neighbour.n\_dist + neighbour.n\_coherence + neighbour.n\_density}{\sum_i neighbour_i.n\_dist + neighbour_i.n\_coherence + neighbour_i.n\_density} \quad ,i = 1,2\ldots K$$

… (4.7)

Once the normalized final score of a representative is found, the coherence value of each representative is multiplied by the normalized score. This indicates the importance of that representative in classifying the given data samples. If the sum of all the Neighbour representative's normalized coherence values is calculated the resultant will be the predicted value (8).

$$predicted\_class = \sum_i (neighbour_i.coherence\_val * neighbour_i.score) \quad ,i = 1,2\ldots K$$

… (4.8)

This approach is inspired from the K-nearest neighbour algorithm [30]. Here K is a hyper parameter that the administrator has to fix based on the nature of the dataset. This phase starts with finding the K nearest representatives of the given data sample. Each representative holds information about their cluster such as coherence value of the cluster, distance between the representative and all the cluster members and number of cluster members that are represented by the given representative. To calculate the score of each attribute of each Neighbour With the given information the normalized score for each of the K nearest representative is calculated using formula

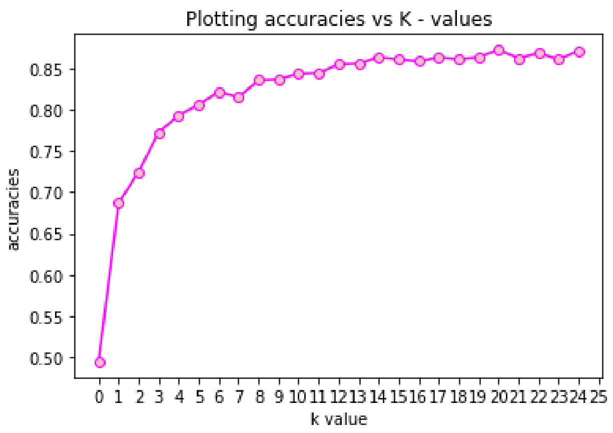


Fig 4.5 Finding optimal k-value using elbow method

The optimal K value is calculated using the elbow method (Fig 4.5) where for each value of k WCSS is calculated [27]. When values of K are plotted with WCSS, from the graph it is observed that 20 is the optimal value as the slope of the curve is coming closer to the x axis. So the conclusion is 20 is the optimal K value for our data samples.

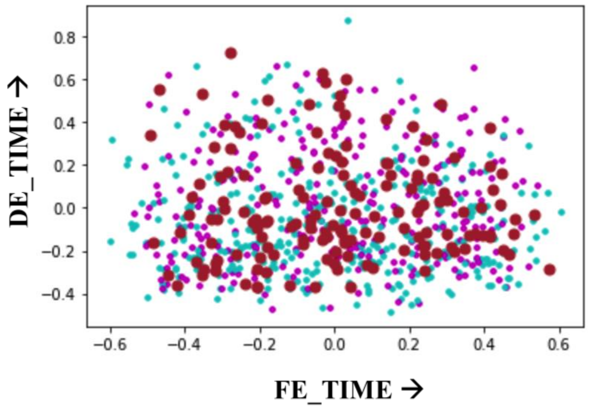


Fig 4.6 scatter plot of the self-organized instances over data samples.

Fig 4.6 shows the resultant representation of the dataset after fixing the representatives and assigning values of attributes to the representatives.

4.2 GENERATIVE ADVERSARIAL NETWORK-BASED DOMAIN ADAPTATION

Domain adaptation in simple terms is the process of drawing relevance between one domain and the other. It is important to consider the fact that drawing resemblance between two indirectly related domains can be quite strenuous and keeping in perspective of different scenarios is also tense. To reduce the cargo of this wide perspective, GAN is used. GAN in short generates its own samples from the existing dataset and discriminates with the real samples to get an accurate prediction. The generator understands how to create the optimal samples to go past the discriminator by employing neural network-based techniques to generate plausible (realistic) samples. The discriminator employs the generated samples as unfavourable training examples. The discriminator trains to differentiate between bogus and true data produced by the generator.

A generator loss is a penalty imposed by the discriminator on the generator for generating unreasonable results. This whole process extending from fake sample generation to generator loss is repetitively done and known as backpropagation. The batch size used in this project lies between 0 – 64 (32). Any value above would result in overfitting.

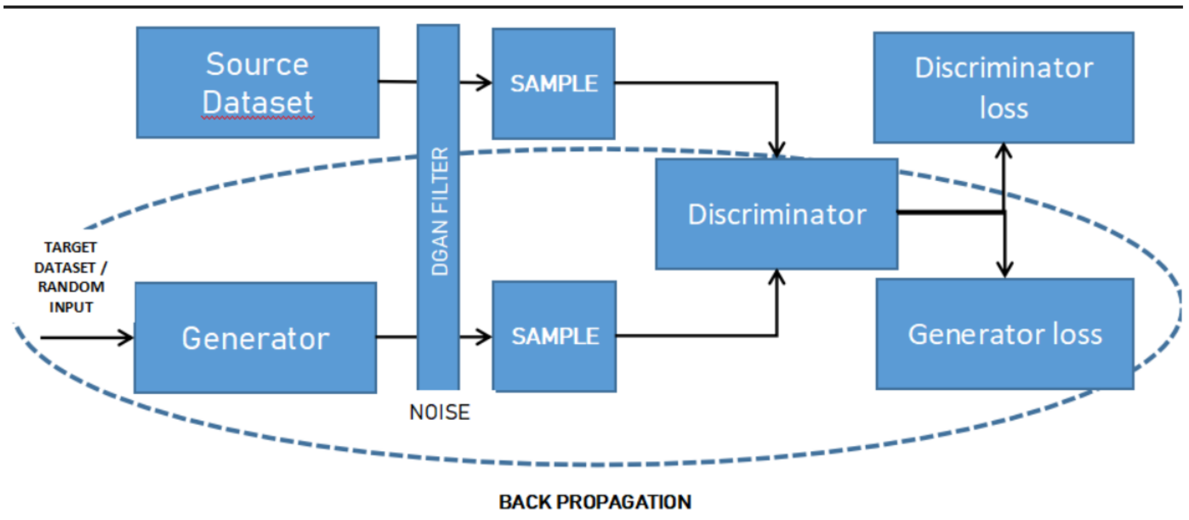


Fig 4.7 GAN Architecture

The above image depicts the working of GAN in our project. Each major component will be broadly explained in the upcoming sub-headings.

4.3 GAN DISCRIMINATOR:

The discriminator receives input from two sources of data, one being the real data consisting of the CWRU dataset and the other being the data generated by the generator. The primary role of the discriminator is to validate whether the generated sample from the generator is equivalent to the

4.4 GAN GENERATOR:

The primary role of the generator in GAN is to create fake samples of the input dataset and make it look as real as possible in order to get past the discriminator. The process of training the generator is more intense than training the discriminator as generator requires tighter integration.

The steps carried out while creating the generator is as follows:

- Random input / Source input
- Generator network which converts random input into plausible sample
- Discriminator network which validates generated data
- Discriminator output
- Loss(generator)

The distribution of the noise generated by the generator doesn't matter much, so it is recommended to choose something that's easy to sample from for instance a uniform distribution. For convenience, the dimensionality space for the noise generated happens to be minute when compared to the space of the output space dimensionality.

(i) **Loss:**

$$L = min[log(d(x)) + log(1 - d(g(x)))] \quad \ldots (4.9)$$

d = Discriminator

g = Generator

L = Loss

The generator here, competes against the discriminator.

Since this loss function and many other correlated functions are bound to be processed more than once the need for reducing the time spent on back propagation is required. The DGAN filter helps in this process by recursively eliminating unnecessary features from the dataset and strengthening the sample so as to avoid some degree of fake sample generation.

4.5 HEALTH INDICATOR CONSTRUCTION:

- Collected multisensory information is used for feature extraction and degradation indicator construction.
- Monotonicity is measured by the difference of positive and negative derivatives and the range is [0, 1]. When the value of Monotonicity is zero, it indicates that the feature F is unmonotonic and when the value is one, it indicates that the feature F is monotonically increasing or decreasing. Then, set a threshold to select only the features that have monotonicity value higher than thaGAt threshold. Here, when the Monotonicity of a feature is at least 8% then that feature will be selected for further processes. Presence of Features with less than 8% doesn't affects the end results. After that use PCA to fuse the features and produce principal components. This normalized PCA1 values are considered as Health indicator values which represents the degradation characteristics of rolling bearings. Then provide the visual description of Health indicator values via plots.
- Here, an exponential degradation model is proposed to mimic the health indicator behaviour. h(t) is the health indicator as a function of time and theta and beta are random parameters deciding the slope of the model.

$$h(t) = \phi + \theta\, exp(\beta t) \qquad \text{.... (4.10)}$$

- To assess the bearing degradation performance in detail during the bearing operation, the baring degradation state is divided into four stages:
    - healthy state
    - slight degradation
    - moderate degradation
    - severe degradation
- Then, validate and compare the HI exponential model with CMPASS dataset and predict the rul.

- Code snippet for performing monotonicity metric on features:

```
mon_df = pd.DataFrame(columns = ['feature', 'monotonicity_val'])

for col in feats:
    mon_val = []
    for unit in df_lag.UnitNumber.unique():
        mon_val.append(monotonicity(df_lag.loc[df_lag.UnitNumber == unit, col]))
    mon_df = mon_df.append({'feature': col, 'monotonicity_val': np.mean(mon_val)}, ignore_index = True)
```

- Code snippet to for fitted exponential behavior of HI modelling:

```
phi = exp_params_df.phi[exp_params_df.UnitNumber == 1].values
theta = exp_params_df.theta[exp_params_df.UnitNumber == 1].values
beta = exp_params_df.beta[exp_params_df.UnitNumber == 1].values

cycles = pca_df.cycle[pca_df.UnitNumber == 1]
pred_ht = exp_degradation([phi, theta, beta], cycles)
```

```
print(pred_ht)

fig, ax = plt.subplots()
sns.lineplot(data = pca_df[pca_df.UnitNumber == 1], x = "cycle", y = "pc1", ax
= ax, label = "True HI")
sns.lineplot(y = pred_ht, x = cycles, ax = ax, color = "green", label = "Fitted
Exponential HI")
ax.axhline(threshold, color = 'r')
ax.text(200,threshold - 0.01,'Failure Threshold',rotation=0)
ax.set_title("Unit: 1")
ax.set_xlabel("Cycles")
ax.set_ylabel("Health Indicator")
```

- Code implementation of RUL of CMAPSS dataset:

```
result_test_df = pd.DataFrame(columns = ['UnitNumber', 'phi', 'theta', 'beta',
'Pred_RUL', 'True_RUL'])
for i in pca_test_df.UnitNumber.unique()
   ht = pca_test_df.pc1[pca_test_df.UnitNumber == i]
   cycle = pca_test_df.cycle[pca_test_df.UnitNumber == i]
   OptimizeResult = optimize.least_squares(residuals, param_1,
bounds=bounds,  args = (cycle, ht, exp_degradation))phi, theta, beta =
OptimizeResult.x
   total_cycles = np.log((threshold - phi) / theta) / beta
   RUL = total_cycles - cycle.max()
      result_test_df = result_test_df.append({'UnitNumber':i, 'phi': phi, 'theta':
theta, 'beta': beta,'Pred_RUL': RUL,
'True_RUL':y_true.RUL[y_true.UnitNumber == i].values[0]},ignore_index =
True)
```

4.6 SUMMARY

Thus to summarize things, firstly the CWRU dataset is preprocessed with the proposed data engineering techniques. Since it’s in unbalanced state, a novel SWOGUT technique is used to sample it. Source equivalent targer datasets had been generated using GAN based architecture and deterioration index is derived on that dataset .Then this index is added as spawned feature to the dataset. Finally the proposed Representation Based Classification model for fault diagnostics is applied on the dataset and the results are discussed in the upcoming chapter 5.

CHAPTER 5
RESULTS AND DISCUSSIONS

5.1 Representation Based Classification

A. Model Evaluation Metrics:

1. Confusion Matrix

Confusion matrix is the performance evaluation metric of a model most widely used in classification problems where there is at most two output classes. So this metric uses the terminology of naming the output classes as either zero or one based on the prediction results and generates a rectangular array representation of original true values vs. predicted values. The formula used to compute the results would appear strenuous as it would include the combinations of truth values.

Table 5.1 Confusion Matrix

	Negative	**Positive**
Negative	TN True Negative	FP False Positive
Positive	FN False Negative	TP True Positive

2. Precision, Recall & F-1 score:

Accuracy is the ratio of summation of the obtained true positive and true negative values to the total predictions made. Precision is the ratio of the total number of true positives to the positive predictions

$$\text{Precision} = TP / (TP + FP) \qquad \text{…. (5.1)}$$

$$\text{Recall} = TP / (TP + FN) \qquad \text{…. (5.2)}$$

$$\text{F1 Score} = 2 * (\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall}) \qquad \text{…. (5.3)}$$

B. Results of RBC with comparative analysis on other existing methods

The performance of the RBC is analyzed by comparing the performance metrics like confusion matrix, Accuracy, Precision, Recall and F1-Score with other existing algorithms such as Support Vector Machine [32], Convolution Neural Network [31] and K Nearest Neighbors [32]. The result shows that the proposed classification algorithm has outsmarted most of the existing classification algorithm with good improvements in all accuracy metrics.

Table 5.2 Confusion matrix for RBC model

	Predicted 0	**Predicted 1**
Actual 0	247859	52651
Actual 1	15342	355435

From table 5.2 it can infer that the propose classification model shows better classification performance by having large number of true positive and true negative.

Table 5.3 Confusion matrix for CNN classifier

	Predicted 0	**Predicted 1**
Actual 0	237859	67451
Actual 1	37342	328635

From table 5.3 it can infer that the CNN classification model shows slightly poor classification performance in comparisons to RBC by having more number of false positive and false negative.

Table 5.4 Confusion matrix for SVM classifier

	Predicted 0	**Predicted 1**
Actual 0	153859	169551
Actual 1	100342	247535

From table 5.4 it can infer that the SVM classifier model has poor classification performance in comparisons to RBC by having large number of false positive and false negative.

Table 5.5 Confusion matrix for KNN classifier

	Predicted 0	**Predicted 1**
Actual 0	209259	90651
Actual 1	65942	305435

From table 5.5 it can infer that the KNN classification model shows less classification performance in comparisons to RBC by having more number of false positive and false negative.

Table 5.6 Comparison of RBC with existing Classification algorithms.

MODEL	ACCURACY	PRECISION	RECALL	F1 SCORE
SVM	60%	61%	46%	53%
CNN	84%	86%	77%	82%
RBC	**90%**	**94%**	**82%**	**88%**
KNN	77%	76%	70%	73%

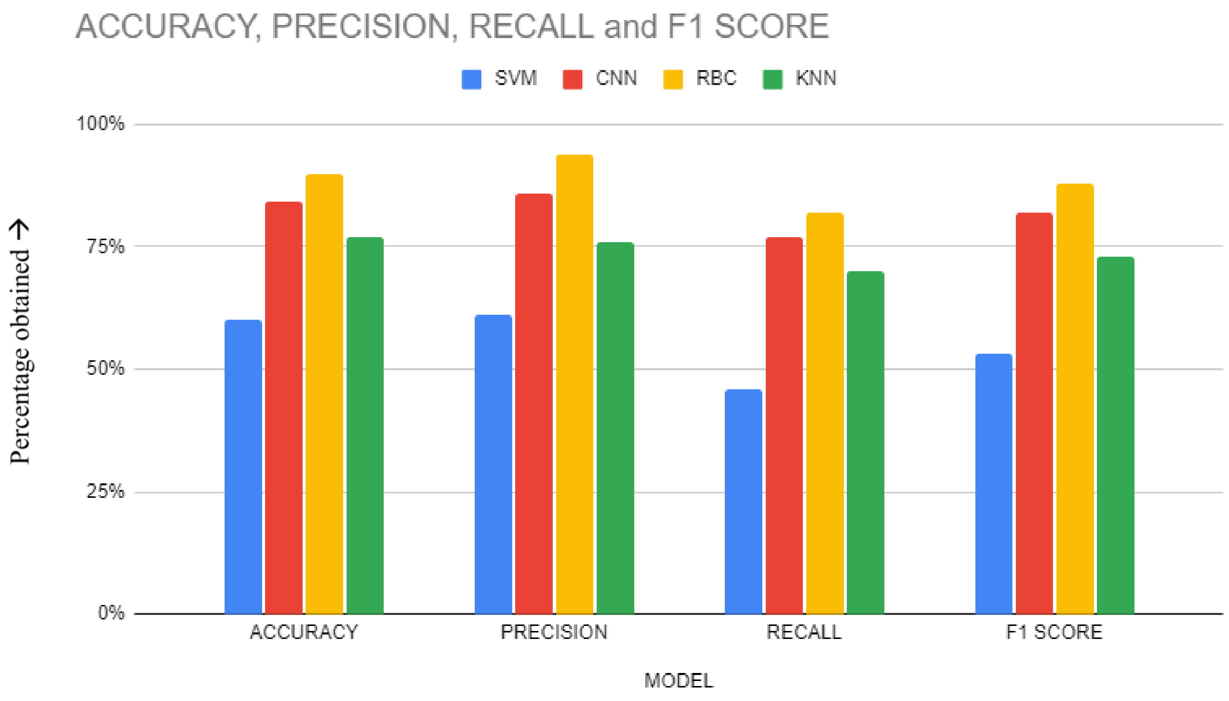


Fig 5.1 Comparison of RBC with existing Classification algorithms.

Based on the comparative analysis from confusion matrices of all popular classifiers and From Fig 3.3, it's obvious that the **RBC** classification model performed better than the other existing classification models.

5.2 HI CONSTRUCTION:

Table 5.7 Calculated loss of TRUE and predicted RUL of CMAPSS dataset

Pred_rul	True_rul	Loss
128.3452	112	-16.3452
122.606	98	-24.606
52.55	69	16.45
70.30	82	11.7

Table 5.7 gives the summary of the compared work, which is a calculated loss between the true rul and predicted rul. Loss percentage is decreased as the model trained with more data, where initially original data is split into four quarters to train and test.

Table 5.8 Comparison of the prediction of RUL of CMAPSS dataset with MSE, MAE, MAPE

Method	MSE	MAE	MAPE
Exp-HI	0.79	0.37	10.97
SVR	0.84	0.48	11.50
NAR-NN	3.52	1.16	33.64
BP-NN	1.64	0.61	14.97
LSTM	1.32	0.70	16.07
GM	1.89	0.77	18.65
ARMA	0.98	0.66	17.38

Table 5.8 gives the error results of Mean-squared error, Mean-absolute error, Mean-absolute-percentage error. From the table, it’s inferred that, HI method was able to give better results with reduced errors. Loss percentage is decreased as the model is trained with more data, where initially original data is split into four quarters to train and test.

Table 5.9 HI compared with RMSE and correlation coefficient

MODEL	Linear model	**Exp model**
RMSE	0.4062	0.2494
Correlation Coefficient	0.9279	0.9762

Table 5.9 obviously signifies that, a non-linear exponential model is better than linear model. Hence, after validating HI, this is then added as a spawned feature to the dataset and using RBC classification, fault diagnosis is done.

The overall the results from various metrics shows that the proposed Representation based classification by deriving heath indicator of data samples has improved classification performance by 10% to 30% then traditional classification models. Also from the results obtained during classification of target dataset, it is evident that the proposed model is capable of classifying the data samples domain invariantly.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.1 CONCLUSIONS:

The study conducted has aimed to i) Since the dataset is highly imbalanced with large ratio of healthy instances over unhealthy instances and the ratio is increasing year by year, it is indispensable to perform dataset balancing and ii). Effective fault prognosis, by developing and describing a classification model that supports prognosis of mechanical components. iii) Compression and Acceleration of the classification model to make it work real-time in edge devices without a considerable amount of latency.

The first goal was achieved successfully with the help of our advanced sampling technique, Synthetic Weighted Over-sampling and Guided Under-sampling Technique (SWOGUT) along with a Python script to automate data collection, combining and organizing data. As SWOGUT follows a heuristic approach to synthesize new data instances, the dataset is well balanced without loss of useful information.

To accomplish the second goal, Representatives based Classification using adaptive self-organizing technique which gave a better result (10% to 30% improved accuracy) compared to most of the existing classifiers was implemented here. The main idea of this approach is to reduce the size of decision boundary [33] to ease the classification process. The initial phase focuses mainly on selecting and fixing the representatives and calculating their coherence value by following proposed constraints. Then the second phase of the algorithm mainly focuses on classification of data instances using the derived coherence values of respective representatives selected using the nearest neighbour approach. Similar to the sampling technique used, the classification model also built such a way that it can be extended to various domains irrelevant to the nature of input data. So it supports all types of classification problems in which data samples exhibits similar neighbourhood properties. The proposed classification model stands out from the other classification model in a way that it allows the administrator to alter the feature space by increasing or decreasing the number of expected coherence values that each representative should have. Thus, with respect to the requirements, the application owner has the ability to tune the performance of the model according to the available resources such as dataset, computational power etc.

Finally, the third goal is to reduce the time and memory complexity of the model and to make it work even in edge devices which require on-demand decisions. This is achieved by eliminating the use of hyper parameters and by using a limited number of trainable parameters. The model is tested against well-known neural networks such as Convolutional neural networks and it is proved that the proposed model showed similar performance with very less resources or complexity.

6.2 FUTURE WORK:

Further the proposed classification model can be extended to various domains such as Bio medical, Cyber security and other domains where the data is not linearly separable and require small decision boundaries to support efficient classification. Instead of binary classification, it can be used to extend it to healthy state prediction by manipulating the dataset. Further the proposed work can be extended to domain invariant classifiers with the help of generative adversarial networks [23]. The performance of the algorithm can be further improved by deriving the health index of the faulty components as an additional feature in the pre-processing stage.

Besides identifying the component failure as a classification problem with only two output classes (binary), it can be extended to a multi classification problem with the introduction of fuzzy inferences in knowledge-based systems. With this approach of multi classification, the degree or extent to which the damage of a mechanical component might occur can be found, which might be a useful metric to predict the remaining useful time of any physical components.

REFERENCES

[1] Y. Ding, M. Jia and Y. Cao, "Remaining Useful Life Estimation Under Multiple Operating Conditions via Deep Subdomain Adaptation," in *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1-11, 2021, Art no. 3516711, doi: 10.1109/TIM.2021.3076567.

[2] Zhao D, Liu F. Cross-condition and cross-platform remaining useful life estimation via adversarial-based domain adaptation. Sci Rep. 2022 Jan 18;12(1):878. doi: 10.1038/s41598-021-03835-2. PMID: 35042894; PMCID: PMC8766616.

[3] N. Kim, H. Lee, J. Lee and B. Lee, "Health management technology for estimating the remaining useful life of household appliances," 2021 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia), 2021, pp. 1-3, doi: 10.1109/ICCE-Asia53811.2021.9642012.

[4] J.Li and D. He, "A Bayesian Optimization AdaBN-DCNN Method With Self-Optimized Structure and Hyperparameters for Domain Adaptation Remaining Useful Life Prediction," in IEEE Access, vol. 8, pp. 41482-41501, 2020, doi: 10.1109/ACCESS.2020.2976595.

[5] Zhang, Yongchao, Zhaohui Ren, and Shihua Zhou. "A new deep convolutional domain adaptation network for bearing fault diagnosis under different working conditions." Shock and Vibration 2020 (2020).

[6] Rai A and Upadhyay SH. A review on signal processing techniques utilized in the fault diagnosis of rolling element bearings. Tribol Int 2016; 6: 289–306.

[7] Qiu H, Lee J, Lin J, et al. Robust performance degradation assessment methods for enhanced rolling element bearing prognostics. Adv Eng Informatics 2003; 17:

127–140.

[8]. Pan YN, Chen J and Li XL. Spectral entropy: A complementary index for rolling element bearing performance degradation assessment. Proc IMechE, Part C: J Mechanical Engineering Science 2009; 223: 1223–1231.

[9] Pan Y, Chen J and Li X. Bearing performance degradation assessment based on lifting wavelet packet decomposition and fuzzy c-means. Mech Syst Signal Process 2010; 24: 559–566.

[10] Yu J. Bearing performance degradation assessment using locality preserving projections and Gaussian mixture models. Mech Syst Signal Process 2011; 25: 2573–2588.

[11]. Ali JB, Fnaiech N, Saidi L, et al. Application of empirical mode decomposition and artificial neural network for automatic bearing fault diagnosis based on vibration signals. Appl Acoust 2015; 89: 16–27.

[12]. Y. Li, M. Xu, R. Wang, and W. Huang, "A fault diagnosis scheme for rolling bearing based on local mean de-composition and improved multiscale fuzzy entropy," Journal of Sound and Vibration, vol. 360, pp. 277–299, 2016.

[13]. Z. Li, J. Chen, Y. Zi, and J. Pan, "Independence-oriented VMD to identify fault feature for wheel set bearing fault diagnosis of high-speed locomotive," Mechanical Systems and Signal Processing, vol. 85, pp. 512–529, 2017.

[14]. Z. Wang, C. Hu, H. Fan, Real-time remaining useful life prediction for a nonlinear degrading system in service: Application to bearing data, IEEE/ASME Trans. Mechatron. 23 (1) (2018) 211–222, https://doi.org/10.1109/TMECH.2017.2666199.

[15]. N. Li, N. Gebraeel, Y. Lei, L. Bian, X. Si, Remaining useful life prediction of machinery under time-varying operating conditions based on a two-factor state-space model, Reliab. Eng. Syst. Saf. 186 (2019) 88–100, https://doi.org/10.1016/j.ress.2019.02.017,URL:http://www.sciencedirect.com/science/article/pii/S0951832018313024.

[16]. Z. Pan, Z. Meng, Z. Chen, W. Gao, Y. Shi, A two-stage method based on extreme learning machine for predicting the remaining useful life of rolling-element bearings, Mech. Syst. Signal Process. 144 (2020), https://doi.org/10.1016/j.ymssp.2020.106899,URL:http://www.sciencedirect.com/science/article/pii/S0888327020302855 106899.

[17]. Lei Y, Niu S, Guo L,Li N. A distance metric learning based health indicator for health prognostics of bearings.In: 2017 international conference on sensing, diagnostics, prognostics, and control (SDPC). IEEE; 2017, p. 47–52.

[18]. Chen Z, Li W. Multisensor feature fusion for bearing fault diagnosis using sparse autoencoder and deep belief network. IEEE Trans Instrum Meas 2017;66:1693–702.

[19]. Chen Z, Guo R, Lin Z, Peng T, Peng X. A data-driven health monitoring method using multi-objective optimization and stacked autoencoder based health indicator. IEEE Trans Ind Inf 2020. http://dx.doi.org/10.1109/TII. 2020.2999323, In press.

[20]. Y. Wang, Y. Peng, Y. Zi, X. Jin, and K.-L. Tsui. A two-stage data-driven-based prognostic approach for bearing degradation problem. IEEE Transactions on industrial informatics, 12(3):924–932, 2016.

[21] Bearing Data Center - “Seeded Fault Test Data”, https://engineering.case.edu/bearingdatacenter.

[22] A. Hanskunatai, "A New Hybrid Sampling Approach for Classification of Imbalanced Datasets," 2018 3rd International Conference on Computer and Communication Systems (ICCCS), 2018, pp. 67-71, doi: 10.1109/CCOMS.2018.8463228.

[23] D. Rusticus, L. Goldmann, M. Reisser and M. Villegas, "Document Domain Adaptation with Generative Adversarial Networks," 2019 International Conference on Document Analysis and Recognition (ICDAR), 2019, pp. 1432-1437, doi: 10.1109/ICDAR.2019.00230.

[24] X. Li, "Remaining Useful Life Prediction of Bearings Using Fuzzy Multimodal Extreme Learning Regression," 2017 International Conference on Sensing, Diagnostics, Prognostics, and Control (SDPC), 2017, pp. 499-503, doi: 10.1109/SDPC.2017.100.

[25] Tiago Zonta, Cristiano André da Costa, Rodrigo da Rosa Righi, Miromar José de Lima, Eduardo Silveira da Trindade, Guann Pyng Li, Predictive maintenance in the Industry 4.0: A systematic literature review, Computers & Industrial Engineering, Volume 150, 2020, 106889, ISSN 0360-8352, https://doi.org/10.1016/j.cie.2020.106889.

[26] Journal of manufacturing systems 56 (2020) 539-557 - Towards multimodal approaches to predictive maintenance A systematic literature survey on diagnostics and prognostics! By Juan José Montero Jimenez, Sébastien Schwartz, Rob Vingerhoeds, Bernard Grabot & Michel Salaüna

[27] D. Marutho, S. Hendra Handaka, E. Wijaya and Muljono, "The Determination of Cluster Number at k-Mean Using Elbow Method and Purity Evaluation on Headline News," 2018 International Seminar on Application for Technology of Information and Communication, 2018, pp. 533-538, doi: 10.1109/ISEMANTIC.2018.8549751.

[28] A. Jezzini, M. Ayache, L. Elkhansa, B. Makki and M. Zein, "Effects of predictive maintenance(PdM), Proactive maintenace(PoM) & Preventive maintenance(PM) on minimizing the faults in medical instruments," 2013 2nd International Conference on Advances in Biomedical Engineering, 2013, pp. 53-56, doi: 10.1109/ICABME.2013.6648845.

[29] M. Pighin and A. Marzona, "Reducing corrective maintenance effort considering module's history," Ninth European Conference on Software Maintenance and Reengineering, 2005, pp. 232-235, doi: 10.1109/CSMR.2005.48.

[30] S. Sun and R. Huang, "An adaptive k-nearest neighbor algorithm," 2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery, 2010, pp. 91-94, doi: 10.1109/FSKD.2010.5569740.

[31] A. B. Husebø, H. V. Khang and W. Pawlus, "Diagnosis of Incipient Bearing Faults using Convolutional Neural Networks," 2019 IEEE Workshop on Electrical Machines Design, Control and Diagnosis (WEMDCD), 2019, pp. 143-149, doi: 10.1109/WEMDCD.2019.8887785.

[32] N. Dutta, U. Subramaniam, P. Sanjeevikumar, S. C. Bharadwaj, Z. Leonowicz and J. B. Holm-Nielsen, "Comparative Study of Cavitation Problem Detection in Pumping System Using SVM and K-Nearest Neighbour Method," 2020 IEEE International Conference on Environment and Electrical Engineering and 2020 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I&CPS Europe), 2020, pp. 1-6, doi: 10.1109/EEEIC/ICPSEurope49358.2020.9160689.

[33] S. O. Al-mamory, "Classification performance enhancement using boundary based sampling algorithm," 2017 Annual Conference on New Trends in Information & Communications Technology Applications (NTICT), 2017, pp. 186-191, doi: 10.1109/NTICT.2017.7976104.

APPENDIX 1

SIMULATION ENVIRONMENT

PYTHON

Python is a high-level language which have been adopted to work in real time development environment to test and work with hard drive failure prediction algorithms using Jupiter notebook and google Collab. Since it correlates much with normal English language it is easier to build and test algorithms in this environment. Hence it helps in writing clear and concise code to put forth our logic to even large-scale projects.

SCIKIT-LEARN

Scikit-learn is a non-proprietary ML library in python. It is designed in such a way that it works in conjunction with other python libraries like Numerical python and Scientific python. It supports many machine-learning algorithms including clustering, regression, classification, decision trees, k-means etc.

TENSORFLOW

TensorFlow is a free open-source library in python. This python library is used for even research purposes even in high level tech companies like MAANG. Tensor flow provides stable python application interface. Due to its easy-to-use interface this library finds its application in many areas of research and development. It is also beginner friendly and hence anyone with keen interest can easily learn it.

KERAS

Keras is an open-source neural network that is built on top of TensorFlow. Keras is written in python. Keras is user friendly, modular, and extensible. It supports both RNN and CNN. It uses activations, optimizers, and objectives. It supports other layers such as batch normalization, pooling and so on.

GOOGLE COLABORATORY

Goole Collaboratory, also called “Colab” is a service provided by google that allows any user to write and execute python scripts through any browser at free of cost. It is mostly suited for analytical purposes. It provides certain memory and GPU restrictions for free users and which can be overcome by using premium services at considerable cost.

APPENDIX II

SCREENSHOTS

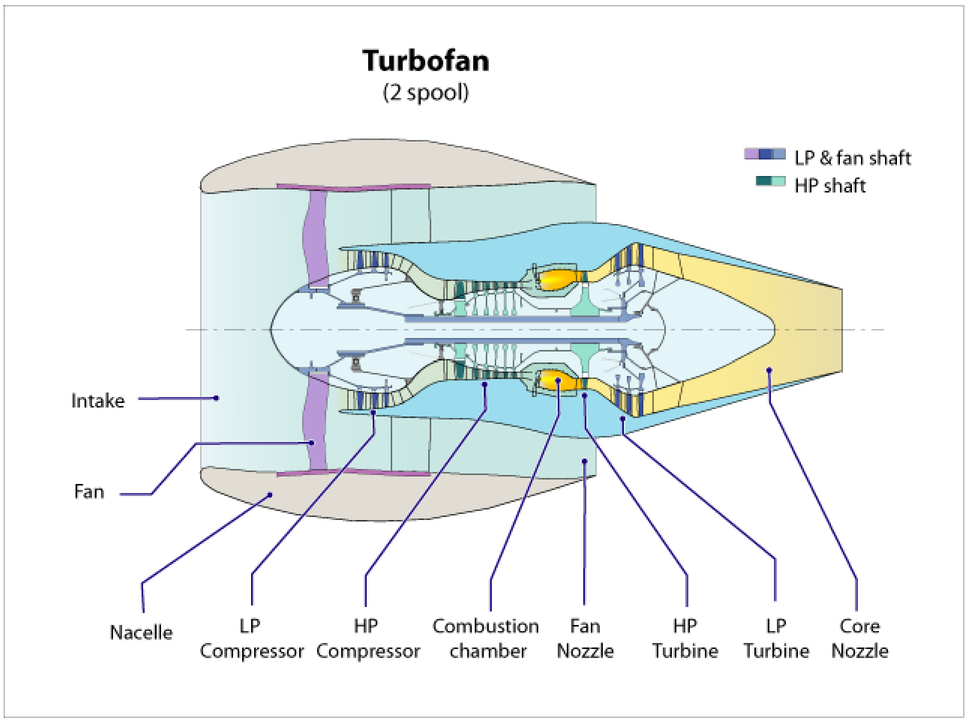
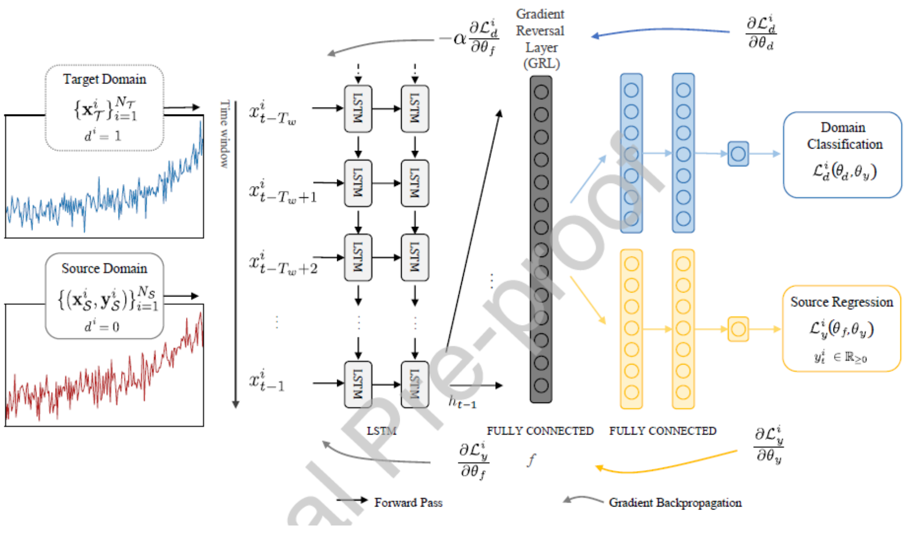
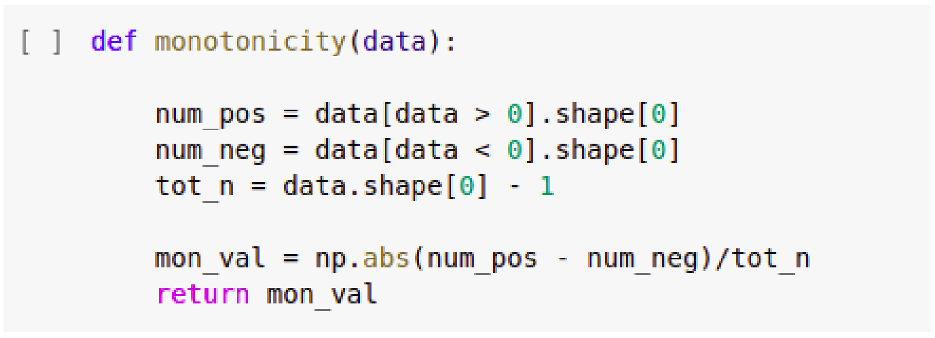


Figure A1 – Parts of Turbo fan



Gradient Reversal Layer (GRL)
$-\alpha \frac{\partial \mathcal{L}_d^i}{\partial \theta_f}$
$\frac{\partial \mathcal{L}_d^i}{\partial \theta_d}$
Target Domain
$\{x_T^i\}_{i=1}^{N_T}$
$d^i = 1$
Source Domain
$\{(x_S^i, y_S^i)\}_{i=1}^{N_S}$
$d^i = 0$
Time window
$x_{t-T_w}^i$
$x_{t-T_w+1}^i$
$x_{t-T_w+2}^i$
x_{t-1}^i
LSTM
h_{t-1}
Domain Classification
$\mathcal{L}_d^i(\theta_d, \theta_y)$
Source Regression
$\mathcal{L}_y^i(\theta_f, \theta_y)$
$y_t^i \in \mathbb{R}_{\geq 0}$
LSTM
FULLY CONNECTED
FULLY CONNECTED
$\frac{\partial \mathcal{L}_y^i}{\partial \theta_f}$
f
$\frac{\partial \mathcal{L}_y^i}{\partial \theta_y}$
Forward Pass
Gradient Backpropagation

Figure A2 – Domain Adaptation Network using LSTM

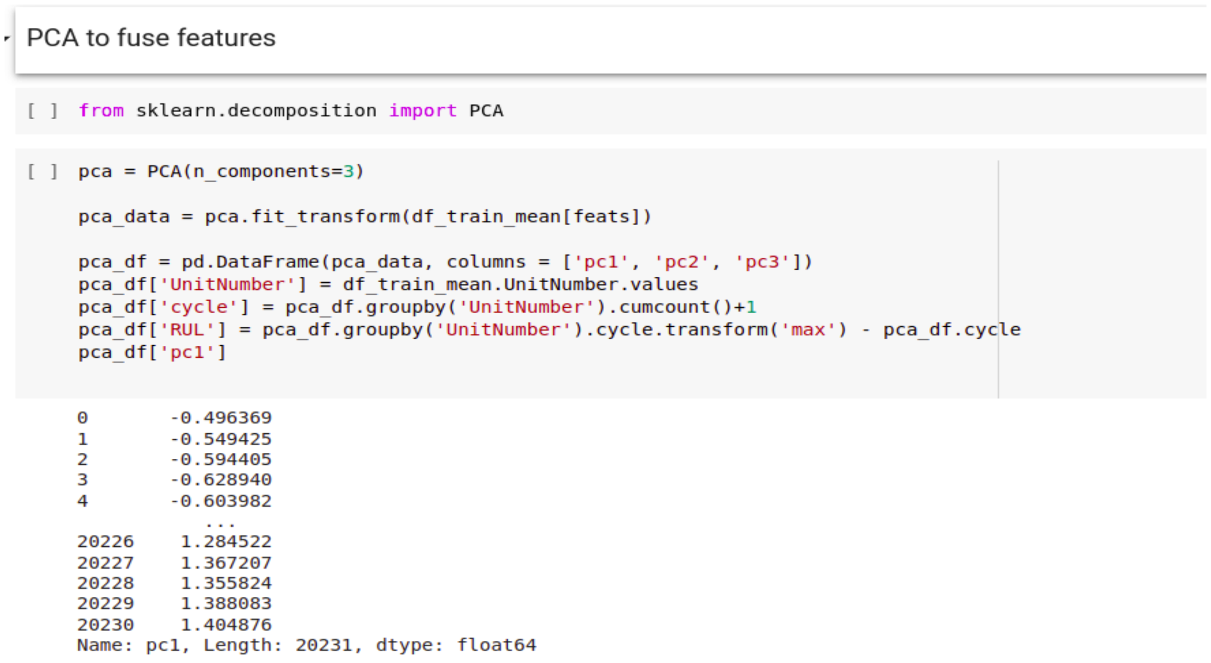


```
[ ]  def monotonicity(data):

         num_pos = data[data > 0].shape[0]
         num_neg = data[data < 0].shape[0]
         tot_n = data.shape[0] - 1

         mon_val = np.abs(num_pos - num_neg)/tot_n
         return mon_val
```

Figure A3 - Code snippet for Monotonicity function:



PCA to fuse features

```
[ ]  from sklearn.decomposition import PCA
```

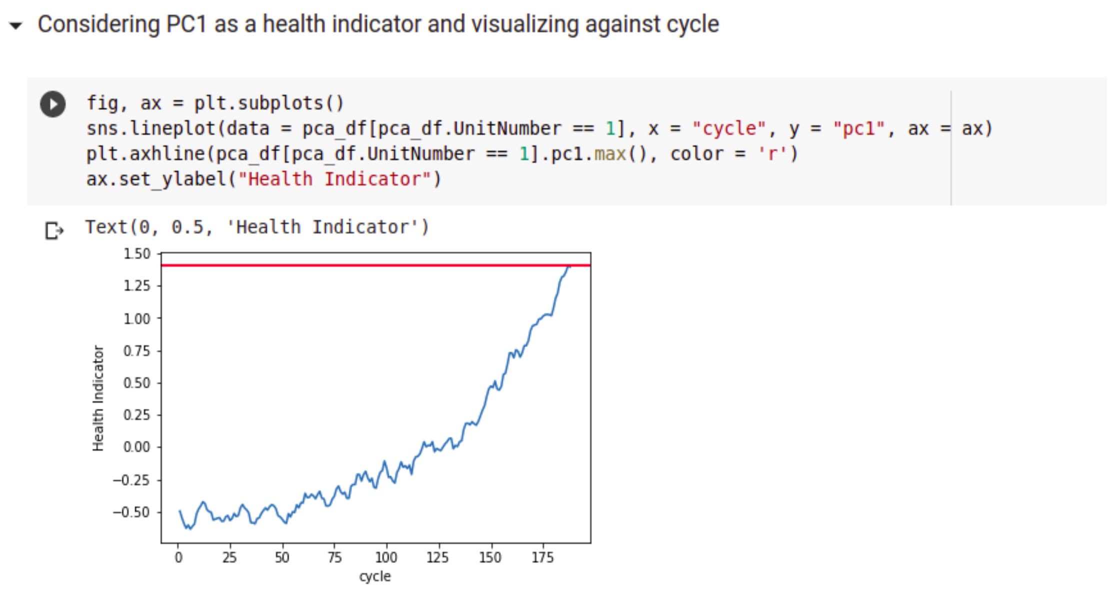
```
[ ]  pca = PCA(n_components=3)

     pca_data = pca.fit_transform(df_train_mean[feats])

     pca_df = pd.DataFrame(pca_data, columns = ['pc1', 'pc2', 'pc3'])
     pca_df['UnitNumber'] = df_train_mean.UnitNumber.values
     pca_df['cycle'] = pca_df.groupby('UnitNumber').cumcount()+1
     pca_df['RUL'] = pca_df.groupby('UnitNumber').cycle.transform('max') - pca_df.cycle
     pca_df['pc1']
```

```
0       -0.496369
1       -0.549425
2       -0.594405
3       -0.628940
4       -0.603982
           ...
20226    1.284522
20227    1.367207
20228    1.355824
20229    1.388083
20230    1.404876
Name: pc1, Length: 20231, dtype: float64
```

Figure A4 - 1-D Health indicator values as a result of PCA



▾ Considering PC1 as a health indicator and visualizing against cycle

```
fig, ax = plt.subplots()
sns.lineplot(data = pca_df[pca_df.UnitNumber == 1], x = "cycle", y = "pc1", ax = ax)
plt.axhline(pca_df[pca_df.UnitNumber == 1].pc1.max(), color = 'r')
ax.set_ylabel("Health Indicator")
```

```
Text(0, 0.5, 'Health Indicator')
```

Figure A5 - Visual representation of HI:

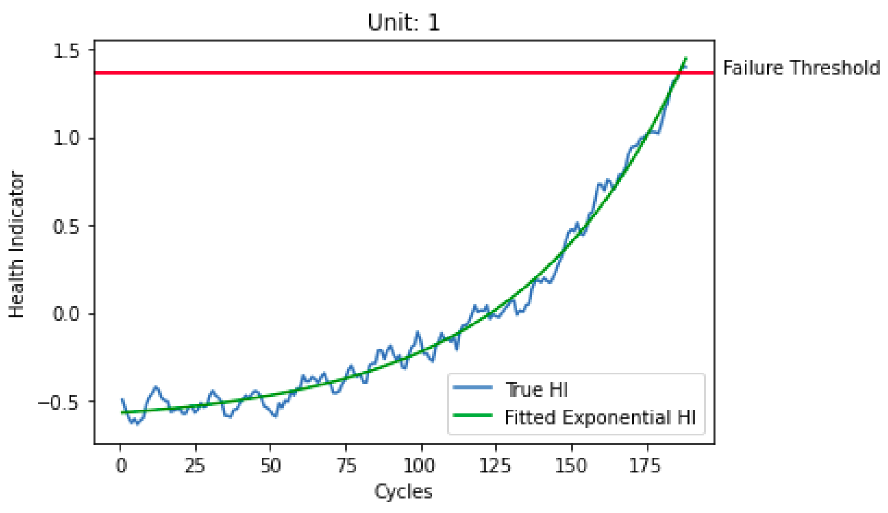
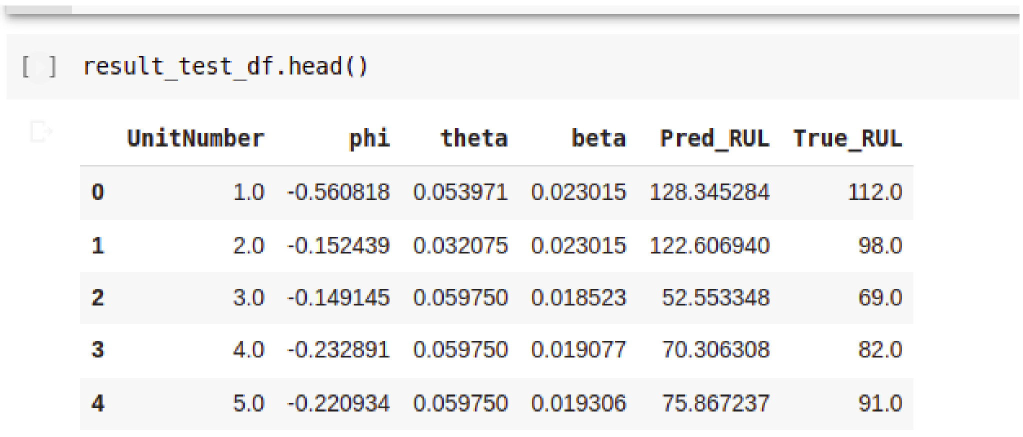


Figure A6 - Visualising the exponential behaviour of the HI



```
[ ] result_test_df.head()
```

	UnitNumber	phi	theta	beta	Pred_RUL	True_RUL
0	1.0	-0.560818	0.053971	0.023015	128.345284	112.0
1	2.0	-0.152439	0.032075	0.023015	122.606940	98.0
2	3.0	-0.149145	0.059750	0.018523	52.553348	69.0
3	4.0	-0.232891	0.059750	0.019077	70.306308	82.0
4	5.0	-0.220934	0.059750	0.019306	75.867237	91.0

Figure A7 - Validated and compared result of proposed HI construction model to predict RUL in CMPASS dataset

```
sns.regplot(x = result_test_df.Pred_RUL, y = result_test_df.True_RUL)
plt.xlabel('Predicted RUL')
plt.ylabel('True RUL')
```

```
Text(0, 0.5, 'True RUL')
```

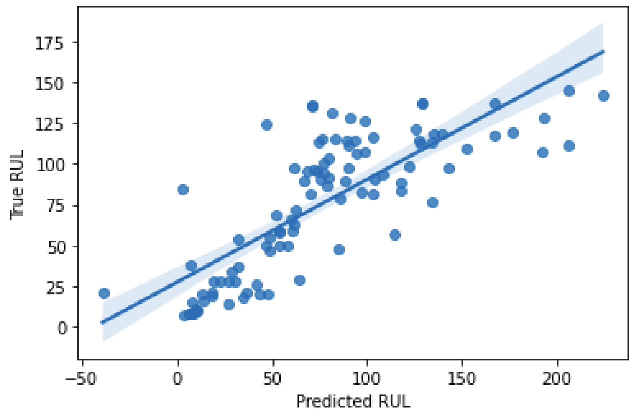
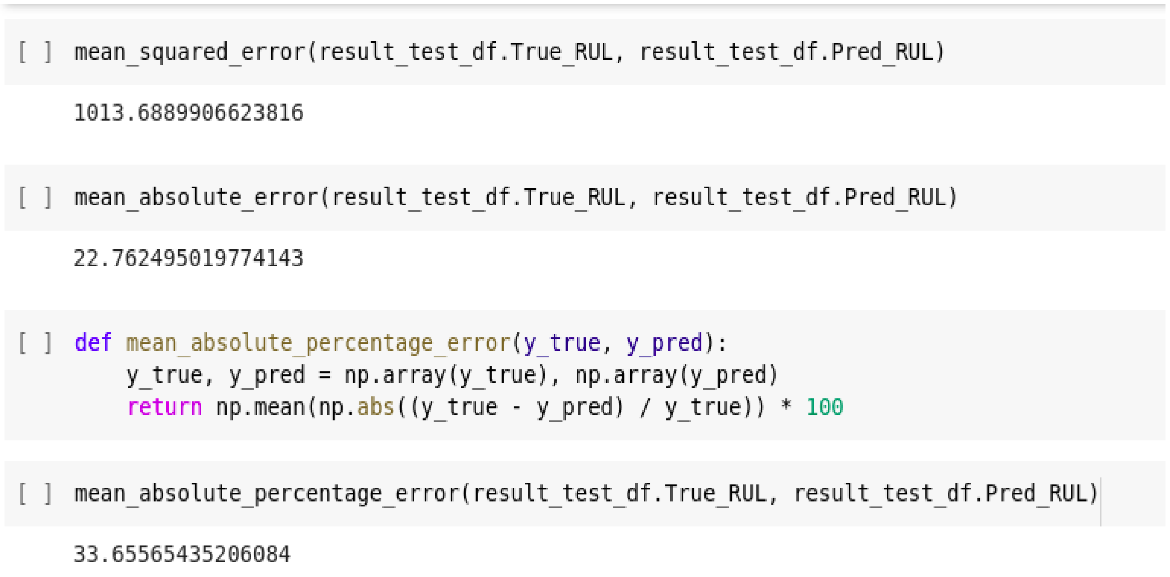


Figure A8 - Visual plot of predicted rul vs true rul of CMPASS dataset



```
[ ] mean_squared_error(result_test_df.True_RUL, result_test_df.Pred_RUL)

    1013.6889906623816

[ ] mean_absolute_error(result_test_df.True_RUL, result_test_df.Pred_RUL)

    22.762495019774143

[ ] def mean_absolute_percentage_error(y_true, y_pred):
        y_true, y_pred = np.array(y_true), np.array(y_pred)
        return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

[ ] mean_absolute_percentage_error(result_test_df.True_RUL, result_test_df.Pred_RUL)

    33.65565435206084
```

Figure A9 - Error results