Artificial Intelligence-based Condition Monitoring of Rail Infrastructure

Wasim AHMAD

Graduation Committee

Chairman and PDEng-program director

Prof.dr.ir. D. Schipper — University of Twente

Thesis Supervisor

Prof.dr.ir. T. Tinga — University of Twente

Daily supervisor

Dr.ir. R. Loendersloot — University of Twente

Company supervisor

D.J. Vermeij MSc. — Strukton Rail

Member(s)

Prof.dr.ing. B. Rosic — University of Twente

Artificial Intelligence-based Condition Monitoring of Rail Infrastructure

Ahmad, Wasim

PDEng thesis, University of Twente, Enschede, The Netherlands

September 2019

Printed by Gildeprint, Enschede, The Netherlands

Cover design by Wasim Ahmad

Artificial Intelligence based Condition Monitoring of Rail Infrastructure

PDEng Thesis

to obtain the degree of

Professional Doctorate in Engineering (PDEng) at the University of Twente,

on the authority of the rector magnificus,

prof. dr. T.T.M. Palstra,

on account of the decision of the graduation committee,

to be defended

on Tuesday the 24th of September 2019 at 13.00 hours

by

Wasim Ahmad

born on 1 April 1990

Mardan, Pakistan

This PDEng Thesis has been approved by:

Thesis Supervisor: Prof.dr.ir. T. Tinga

Co-supervisor: Dr.ir. R. Loendersloot

Summary

The design cycle of the rail condition monitoring system (CMS) consists of problem investigation, treatment design, and validation. The aim of the project is to improve the rail maintenance by timely reporting the incipient rail defects. On-time and appropriate maintenance results in reduction of maintenance cost, short down-time and high service availability. A massive data has been collected from railway system through various sensors but the connection between the sensors data and the rail condition is not known. Moreover, currently the maintenance strategies are triggered mostly based on human inspection. Therefore an automated rail CMS need to be developed that makes intelligent decisions using the data and helps in initiating a timely maintenance process. The rail defects need to be detected at their earliest stages which could otherwise lead to severe defects and cause rail failure. Therefore the aimed system will help in carrying out predictive maintenance of the rail infrastructure. The detailed description of the problem is discussed in chapter 1. The solution for the design problem is build upon the need and requirements of the stakeholders and system. Everything that the stakeholders expect from this solution is included in the list of requirements. Moreover, requirements at the system level are also identified and aimed to be achieved. A comprehensive list of requirements is given table 2.1 in chapter 2.

The design of the rail CMS is based on the train axle box acceleration (ABA) data, that is used by the machine learning (ML) pipeline for information retrieval about rail condition. The designed ML pipeline for rail CMS is illustrated in figure 3.1 of chapter 3. The pipeline consists of ABA pre-processing, extraction of features by using time domain analysis, and anomaly detection algorithm for detecting irregular patterns in ABA. The algorithm for anomaly detection is presented in detail

in chapter 4. The validation process is based on the comparison of the actual rail defects and anomalies detected by the algorithm in ABA data. The flowchart for the validation process is shown in figure 5.1 in chapter 5. Video images of rail infrastructure are utilized for performing the validation process. The visible rail defects in the images are manually labelled and feed to the validation model for comparison. The performance metrics such as hits, mishits and false alarms etc. are calculated using the validation model. The design and user guide for the graphical user interface (GUI) of rail CMS is covered in chapter 6 that explains various components in the layout and discusses the inputs and outputs of the system. Finally the discussion, conclusions, and recommendations are presented in chapter 7 of the thesis report.

Acknowledgements

Firstly, I am grateful to Almighty ALLAH, who blessed me with the physical and intellectual power to accomplish the task of the thesis writing, which is a partial requirement for the PDEng degree. I offer my gratitude to my dear parents who guided and supported me morally and financially at every stage of my life to achieve this success. Moreover, I appreciate the kind and affectionate supervision of my supervisors Prof.dr.ir. Tiedo Tinga and Dr.ir. Richard Loendersloot, that enabled me to complete my degree in a sophisticated way and instilled in me profound analytical skills and also helped me by giving guidelines regarding thesis writing. I am also thankful to Anthonie Boogaard (my colleague at Strukton Rail) who helped me in understanding the datasets, required for this project. I extend my sincere gratitude to all my family members, friends and colleagues at Dynamics Based Maintenance (DBM) Lab who's well wishes and support helped me to achieve this milestone.

Wasim Ahmad

Netherlands, Sept 2019

Contents

List of Figures

List of Tables

List of Abbreviations

CMS	Condition Monitoring System
ABA	Axle Box Acceleration
EC	Eddy Current
US	Ultra Sonic
CBM	Condition Based Maintenance
SVM	Support Vector Machines
LOF	Local Outlier Factor
AUC	Area Under the Curve
KPI	Key Performance Indicator
UT	University of Twente
AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
RMS	Root Mean Square
PCA	Principle Component Analysis
SVD	Singular Value Decomposition
INCOSE	International Council on Systems Engineering

Dedicated to my dear Parents, who's upbringing, guidance, support and peerless love enabled me to achieve this success...

Chapter 1

INTRODUCTION

1.1 Background and Motivation

Train service is one of the most convenient and reliable transportation sources these days. The quality of the railway service is measured based on the train punctuality and service availability. The railway network in The Netherlands is around 2800 km long that includes 6500 km of tracks and 4700 km of electrified tracks. The network contains 4500 bridges and tunnels, 8700 switches, 3000 level crossings, and 380 stations. It serves more than 1200000 passengers per day using 6000 trains. Sometimes delays and interruptions occur in the train service which are most often caused by erupted issues on railway network and requires maintenance to resume the service. The maintenance cost for only squat-related rail defects exceeds 5000 euro/km in a year in Dutch railway network because it is one of the most intensively used network in Europe. Therefore, avoiding the disturbance in train service is highly important not only due to high maintenance cost but also the service delays and downtime is highly unwanted to the passengers.

A fatal train accident occurred at Potters Bar, England On 10 May 2002. The accident took away the lives of seven people and more than 70 people were injured.

The causes of the event were found out to be defects on rail and inadequate maintenance. In order to maintain a safe and uninterrupted train transportation service, appropriate and timely maintenance activities need to be done (Veit, 2007). It is a challenge to determine the right time when these maintenance activities should be performed. In figure 1.1, a PF-curve is given which shows the maintenance techniques with the passage of time depending on the system's condition. The maintenance strategies such as reactive maintenance and preventive maintenance are triggered depending on the time of defect detection. Detection of rail defects at severe condition needs replacement of rail components, however in case of minor defects, grinding and milling is required to stop the defects growth. Artificial intelligence (AI) and machine learning (ML) techniques need to be involved to detect rail defects at the earliest stage of degradation where the resistance to failure is still high.

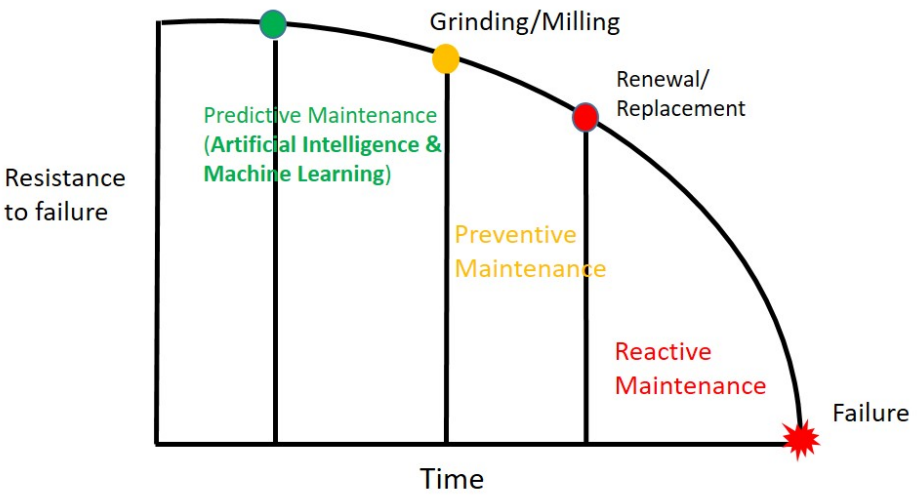


FIGURE 1.1: PF-Curve illustrates maintenance techniques that trigger various maintenance strategies

1.1.1 Railway Maintenance

Reactive maintenance has been performed in response to rail failures in order to resume the train service, however it is considered as too late and can cause fatal train accidents. It does not keep the rail in optimal condition, minor defects are not fixed on time that deteriorates the rail condition fast, hence results in shorter asset life expectancy. Usually preventive maintenance (PvM) is carried out by most of the companies to maintain the rail network intact. However, it is always hard to come up with a perfect maintenance scheduling policy. An appropriate frequency of maintenance need to be settled that is neither too short nor too long. Shorter length and too frequent maintenance could result in rail traffic disturbance and high maintenance cost. On the other side, keeping maintenance interval very long could bring systems failure that is highly undesirable. These failures halt the train service with an undetermined down-time and the rail maintenance companies suffer huge economic losses. On top of everything, these delays in train service bring inconvenience to passengers. Moreover, various other methods have been used for railway condition monitoring (Magel et al., 2008). The methods currently used for rail health monitoring in The Netherlands are visual inspections and eddy current and ultrasonic measurements (Thomas, Heckel, and Hanspach, 2007). However, methods like these are more efficient at severe stage of rail degradation and not regarded as optimal. Furthermore visual inspections are hard to carry out and time consuming, and more importantly the outcome of the inspections rely on the human operator which could be erroneous (Marino et al., 2007). Predictive maintenance for rail infrastructure is imperative in covering the limitations in other approaches for maintenance.

1.1.2 Design Challenge

The detection of rail defects at its earliest stage is paramount to keep the rail in good condition. The existing rail monitoring techniques cannot detect the incipient rail defects that grow later into severe defects. Therefore, an automated and intelligent rail CMS needs to be developed that is capable to identify the early stage defects on rail surface. Development of such a system is vital to efficient and robust rail infrastructure management because it can trigger an appropriate maintenance process at the right time. Intensive effort and work is already going on to develop physics-based models for railway maintenance, however it takes long time and still difficult to implement. On the other side, the availability of huge amount of sensors data provides the opportunity to develop a data-driven model for rail health monitoring. The ABA data has been used by Dutch railways for defect detection such as corrugation and poor quality welds since the mid-1980s (Esveld, 2001). The main advantage of ABA compared with other methods is its lower cost and ease in maintenance. The employment of AI techniques on ABA data can reveal useful information about the condition of the rail system.

1.2 Objective and Scope

In most cases, sensors data are used for data-driven condition monitoring systems. Similarly for railway infrastructure, the sensors data can be utilized and meaningful information can be extracted to reveal rail condition. The sensors, particularly accelerometers, are installed on the axle-box of the train which measures acceleration of the axle-box when train rolls over the rail. The patterns in ABA signal

change with rail anomalies. AI and ML techniques are renown for extracting meaningful insights from sensors data that could be interpreted as understandable information for humans i.e. maintenance personnel. Using AI, the ABA data will be transformed into management data and human understandable information that would help in decision making for rail maintenance. The development of an AI-based application using ABA aims to detect the incipient rail defects that does not require renewal and replacement of the rail assets. Moreover it will not allow the rail defects to reach severe condition that ultimately prevents rail failure and train service derailment. This project focuses on development of data-driven condition monitoring of rail assets by addressing the following design problem:

"Applying the signal processing techniques and ML algorithms to extract meaningful insights from ABA data and detect abnormal patterns in it. These patterns represent irregularities on rail surface." The final deliverable of the project will consist of a ML pipeline that can be operated using the designed graphical user interface (GUI).

1.3 Approach

Based on a literature study and regular meetings with stakeholders and project supervisors, the requirements and constraints of the design task are determined, which will be presented in chapter 2 of the thesis report. Information collection is important to start the activities of product design. The stages related to the aimed project consist of three steps considering the design cycle, see figure 1.2. The first step in the design cycle covers the problem identification, stakeholders and goals. The 2nd step of the design cycle is the design phase that deals with the requirements

and problem solution, and the 3rd step is validation of the solution by comparing the the predicted outcome with true outputs.

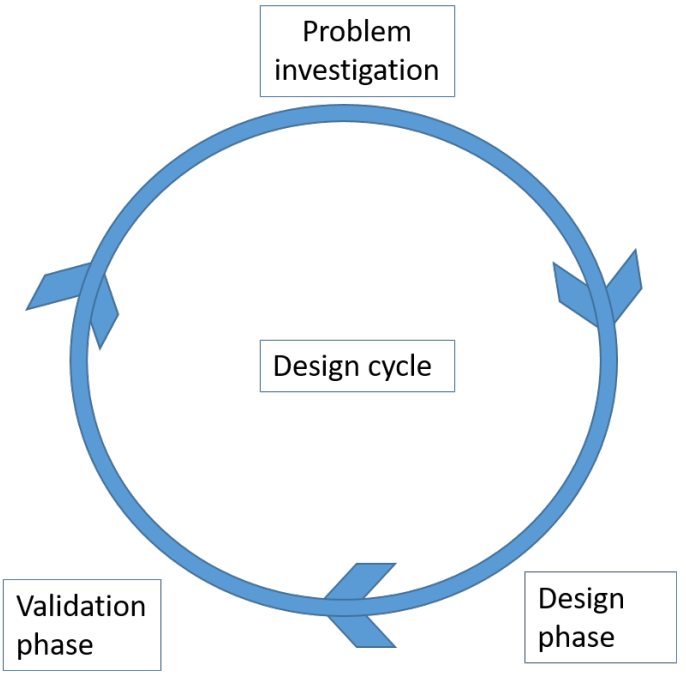


FIGURE 1.2: Design cycle

The feedback from the validation step is used to adjust the parameters of the data-driven condition monitoring model. The model is optimized after a sufficient amount of iterations. Figure 1.3 shows such a feedback system for designing the tool. According to the diagram: (i) If the design is promising, it is adjusted iteratively, (ii) In case the design requires a lot of changes to meet the requirements and needs of the project, a new design is planned, (iii) once the design reaches optimal state, it is regarded as acceptable design. The more iterations it performs, the more accurate the model becomes. The model accuracy cannot be measured directly during the validation process in this case because there is no absolute output

available. However confidence on the data-driven model and the data can be built, if the reported anomalies in ABA data represent rail abnormalities. For verification of the detected anomalies, synchronized video camera images of the rail will be utilized.

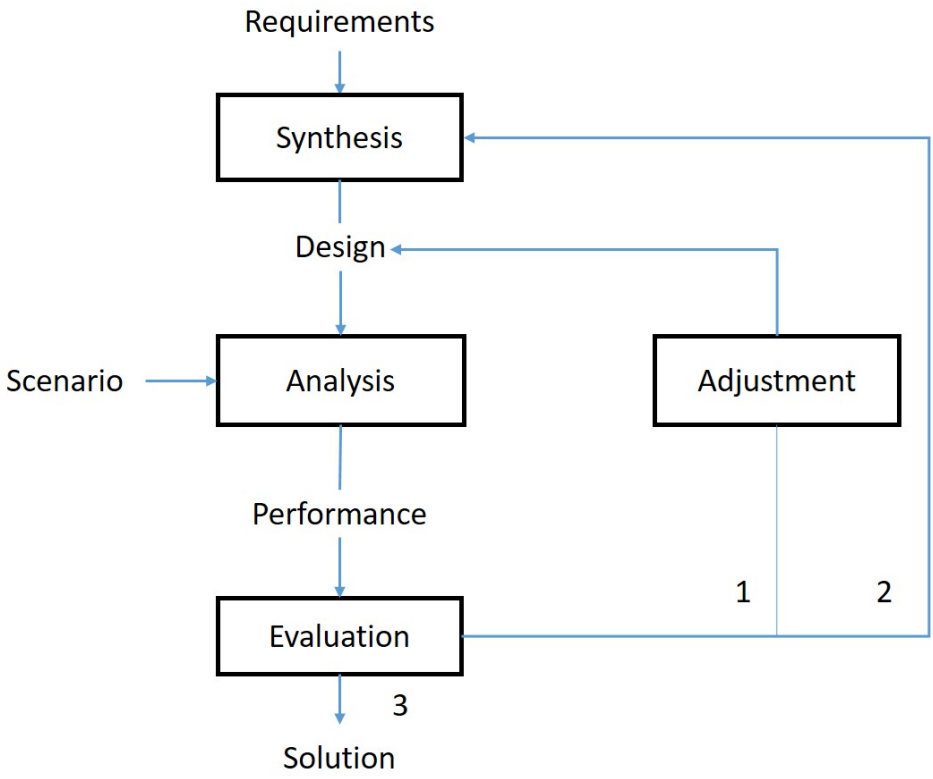


FIGURE 1.3: Iterative feedback mechanism for solving a problem

The aimed rail CMS, from the value engineering perspective, should enhance the quality of the product while reducing the cost and time. Value engineering according to Wikipedia is defined as "A systematic method to improve the *value* of goods or products and services by using an examination of their function". Value, as defined, is the ratio of function to cost and can therefore be manipulated by

either improving the function or reducing the cost. The current design project is aiming to improve the efficiency of decision making for rail maintenance. It attempts to achieve a high level of reliability by monitoring rail condition and reporting the detected defects. Improving the product reliability will significantly reduce the maintenance time and cost ultimately.

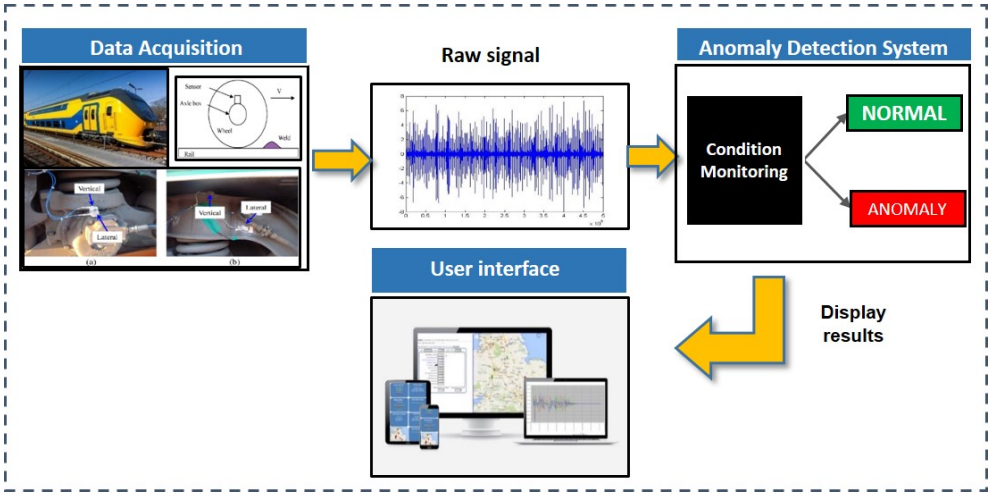


FIGURE 1.4: Overall picture of the data-driven rail condition monitoring system

1.4 Thesis Outline

The PDEng thesis report covers the details of development of rail CMS in seven chapters in total. Chapter 2 provides details about the system's and the stakeholders' needs and the transformation of those needs into requirements. The overall picture of the rail condition monitoring system (CMS) is shown in figure 1.4, which consists of data acquisition, anomaly detection and graphical user interface (GUI) design. The task of data acquisition and initial data processing are performed by Strukton Rail. The anomaly detection part consists of three steps, in which the first

two steps pre-processing and features extraction will be discussed in chapter 3. The setup for data acquisition, sensor types, positioning and dataset description is also presented in this chapter. Details of the implementation of the anomaly detection model are explained in chapter 4. Chapter 5 of the thesis report, provides validation and analysis of the model's results while the design of the graphical user interface (GUI) for the rail CMS is presented in chapter 6. Finally, chapter 7 provides the discussion and the conclusions drawn from the entire process and indicates the future possibilities of research and development in the data-driven maintenance of rail infra-structure.

Chapter 2

Requirements Engineering

2.1 Introduction

In the process of product designing, it is as important as the product itself to determine what the anticipated needs are from the project and how to transform these needs into requirements. In an earlier phase of design process, a preliminary design is made based on the very little information available where a lot of uncertainties exist. But with the passage of time as more and more information is collected, the requirements develop which leads to a clear picture of the design product. It is easy to change things in the preliminary design phase but as the design process goes ahead, the uncertainties in project and ease of change decrease, on the other hand the committed-to requirements increases (Bonnema, 2014). Once the requirements were completely refined and there was no ambiguity regarding project objectives, a final solution design is made. The most important thing for any project is the identification of stakeholders, knowing the people who are directly or indirectly involved in the project. Strukton Rail is the sole stakeholder of this project. Multiple meetings with stakeholder were conducted to determine their needs and expectations from this project. The needs of the stakeholder lead

to project requirements. The project requirements are determined and matched with the design challenge, stated in chapter 1. The purpose of the requirements engineering is to justify stakeholder's needs and find out whether the requirements are feasible to achieve within the allocated time and scope of the project. Requirements engineering is considered a common concept in systems engineering that entails the process of discovering, developing, and tracing, analyzing, qualifying, communicating and managing requirements that define a system at successive levels of abstraction.

2.2 Stakeholder

Strukton Rail is the one and only stakeholder of the data-driven rail maintenance project. It is a multi-national company that focuses on transport systems in densely populated areas, creating access to mining and port areas, and transportation of energy. Strukton is also working on spreading rail tracks in different areas across Europe and outside. They are putting efforts to bring in the state-of-the-art techniques to improve rail maintenance scheduling and reduce the cost. A huge amount of data is available at Strukton in the form of the train ABA, eddy current (EC) and ultra-sonic (US) measurements. Strukton is interested in utilizing the available big data to enhance the rail maintenance strategies.

2.3 Requirements engineering and management

There are several levels of needs and requirements for a project according to the international council on systems engineering (INCOSE) guide. The first level where enterprise strategies are expressed in the form of needs is called "enterprise" level.

Other four levels i.e. the business management level, the business operations level, the system and the sub-system level describe how the needs are transformed into the project requirements. The needs at the enterprise level and system/sub-system level are identified for rail CMS.

2.3.1 The enterprise Level

According to (ISO/IEC/IEEE29148, 2011), the operational concept explain the function of the system (what) and the reason why the system is performing the function (why). The enterprises involved in this project are Strukton and UT which are working in collaboration to develop a data-driven rail maintenance system. In this project, the big data acquired through various sensors and devices are utilized for development of the data-driven system. Maintenance optimization and cost reduction are the identified needs of Strukton at this level of needs. The enterprise level covers the strategies for the rail CMS as follows:

- What: Enhance condition monitoring and defects detection for rail system by transforming the approach of human based visual inspections to an automatic and smart inspection technique.
- Why: Improve the decision making related to rail maintenance in order to reduce maintenance cost.

2.3.2 The System/Sub-system Level

The system/sub-system level where the selection methodology is defined in physical and logical views, is usually used for converting the needs and requirements of Strukton Rail into needs and requirements of the aimed system. These levels shall fall in solution domain where the respective system needs and requirements are

defined. The focus of system/sub-sysem level is on how the rail defect detection could be improved using the rail CMS. The goal of rail CMS at the system level is to detect the abnormality on rail at its precise location, hence accuracy and reliability of the system is highly important. The system is supposed to run in python environment on any computer operating system. The development of the tool shall be done in such a programming technology that is compatible for integration with the Strukton main condition monitoring system. The target system shall be flexible in order to get updated for functionality improvement and fixing programming bugs. The system shall provide a user-friendly interaction to the maintenance engineer/operator. The needs at the sub-system level are what the systems components and functions require for its operation. These needs include, a sufficient storage capacity to hold the big data and store the systems results, in case cloud storage service is not available. Besides that, a sufficient memory and a high processing capability is vital at the time of systems operation, otherwise it takes longer time to process the data and at times the program get crashed.

2.3.3 List of Requirements

Various types of requirements can be identified for a project as mentioned above. However, for rail CMS, the requirements at the stakeholder- (SH) level and system (SYS) level are defined. Stakeholder requirements answers the questions such as "What should be done?", "How well should it be done?" and "Why is it done?", all these questions are related to the enterprise level, and the latter defines the solution and provides an answer to the question "How is it solved?" at the system level. A detailed overview of both types of requirements is provided in table 2.1.

TABLE 2.1: Stakeholder and system level requirements

Type	Label	Description
General	SH1	The developed tool shall be operated by the maintenance operators with no or limited technical knowledge of underlying data analytics
Applicability	SH2	The developed maintenance system shall be applicable for condition monitoring of various rail tracks
Reliability	SH3	Predictions about rail health condition shall be reliable to improve the maintenance strategies
Readability	SH4	The software shall provide operators with enough information when it makes a decision i.e. location and severity of the anomalies
General	SYS5	The deliverable shall be presented in the form of a condition monitoring system (CMS). A software with machine learning algorithms working at the backend of graphical user interface
	SYS6	Python shall be used as programming language to develop the software and all its algorithms
	SYS7	Systems hardware with high processing power are required to run the CMS
	SYS8	CMS shall be user friendly and self-explanatory for operators to use
	SYS9	The tool shall operate on ABA data only as an input, data need to be pre-processed before performing anomaly detection
	SYS10	The software shall save outputs in a database that can be used in future for performing trend analysis in the data
	SYS11	The maintenance software shall visualize the outputs in various ways through plots and enlist detected defects with their geo-locations
Readability	SYS12	The programming code shall be well written and properly commented to provide a sound understanding for developers
Maintainability	SYS13	The rail CMS shall be accessible to developers at Strukton for updates and bug fixing
Scalability	SYS14	The software shall be scalable and robust to dataset size

2.4 Requirements analysis

Project requirements are verified through regular consultation with people at Strukton Rail. The feasibility of these requirements has been tested and validated after discussion, experiments and analysis.

2.4.1 Verification and Testing

The identified requirements are refined and verified through iterative meetings with supervisors at University of Twente and Strukton Rail. The requirements were discussed and considered as valid and practical by project manager at Strukton. Some of the requirements that were over-ambitious and hard to achieve in the available time, were removed from list of requirements. The principles regarding identifying and writing these requirements were thoroughly followed according to INCOSE guide for writing requirements. The requirements are also validated by implementing these in the design project. They define a set of goals and boundaries for developer. The developer needs to stick to the requirements to achieve the goals while staying inside the restricted boundary. The time duration, available resources and feasibility of the actions are to be considered while working on the project in its design phase. Requirements gathering is performed usually in the earlier steps of the design cycle of a project, however it has an impact on every stage of the design cycle. These requirements can be adapted during the development process.

The VEE-model shown in figure 2.1 explains how these requirements can influence various phases of the project during its development. Moreover, it tests the compatibility and validation of the requirements at different phases of the project. Part of the requirements are verified by consultation with experts, some of these are

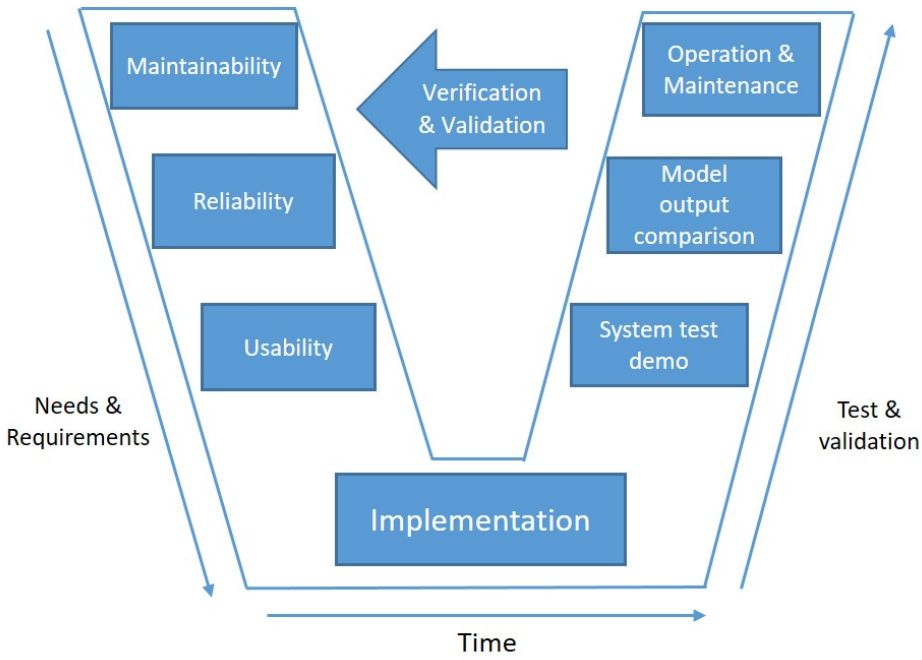


FIGURE 2.1: Vee-Model for intelligent rail condition monitoring system

validated during development phase and others after design implementation. The VEE-model illustrates the impact of requirements and some necessary tests for rail CMS tool. To the left of the VEE-model, the project requirements are given while the right side of the model presents the corresponding verification tests. These tests conform whether the required needs are met or not. This model demonstrates a vital role of the requirements during the design life cycle of the product. The VEE-model tells about the right time and appropriate way to test these requirements. Some of the requirements are easy to be verified, however there are requirements that need specific tests to be checked. For example SH1, requires a prototype of the CMS to be operated by non-technical maintenance operators. Their response can serve as a feedback for CMS modification and improvement. Some of the examples

from the list of requirements are mentioned in the given model.

2.4.2 Risks

The potential risks involved in this data-driven project were the available time for the system development and applicability of ABA for condition monitoring. However a well-organized schedule for all tasks in the project mitigated the time risk. Moreover, the risk related to the type of data (ABA) was also not threatening to halt the project, because the yielded output of the data-driven model is promising. Thus, there was no extra-ordinary risk that hurdled the development of the ABA-based rail CMS.

2.4.3 Performance Indicator

The implementation of the project design, that will be explained in coming chapters, reveals that the key performance indicator (KPI) of the rail CMS is its capability of detecting any sort of abnormality in ABA data. The reported anomaly by the rail CMS can either be an accurate detection of a rail defect or a false-alarm when compared with the ground truth. In other words the performance metrics such as accuracy, false alarms, hit-rate and mishits are considered to be the KPIs of the system.

2.5 Conclusion

The stakeholder needs and requirements are identified and transformed into the system/sub-sytem level requirements. The requirements can be adapted during the development process. The validation and testing of the requirements can either be performed by consultation and discussions with stakeholder or by following the

testing approach in the VEE-model. The risks anticipated at the earlier stage of this project was the time for implementing the design project and feasibility of ABA data. Both of these risks are mitigated by proper task scheduling and the model design of the rail CMS. The outcome of the system is interpreted in the form of performance metrics such as hits and false alarms etc. The metrics are regarded as the KPIs for the system.

Chapter 3

Design of Rail Condition Monitoring System

3.1 Introduction

The design of the rail CMS is based on the accelerometer's data obtained from the axle box of the train. When a train runs over the rail, the axle box in the train vibrates with a certain level. A change in the vibration would occur if the train experiences any irregularity on the rail while running over it. This unusual behavior could be aroused because of various factors i.e. rail defects, objects, rail misalignment, train wheel fault, sleepers etc. The aim of the rail CMS is to catch these anomalies in the ABA data. Anomalies are data patterns that have different data characteristics from normal data patterns. The detection of anomalies has a huge significance and often provides meaningful and critical information in various application domains that requires an immediate action. The ABA data in its original raw form is quite complicated and do not reveal meaningful insights about rail condition. That is why the data is pre-processed and the statistical features are extracted from the data which are used as input to the anomaly detection technique. The anomaly

detection technique separates outliers from the normal data. These anomalies are further analyzed to find their location on the track and severity etc. The 2nd step of the design cycle, which is implementation of the solution, consists of three main steps: (i) data pre-processing, (ii) feature extraction and (iii) anomaly detection. The pipeline given in figure 3.1 illustrates all these three phases of the implementation. However this chapter covers the data pre-processing and feature extraction part of the overall methodology.

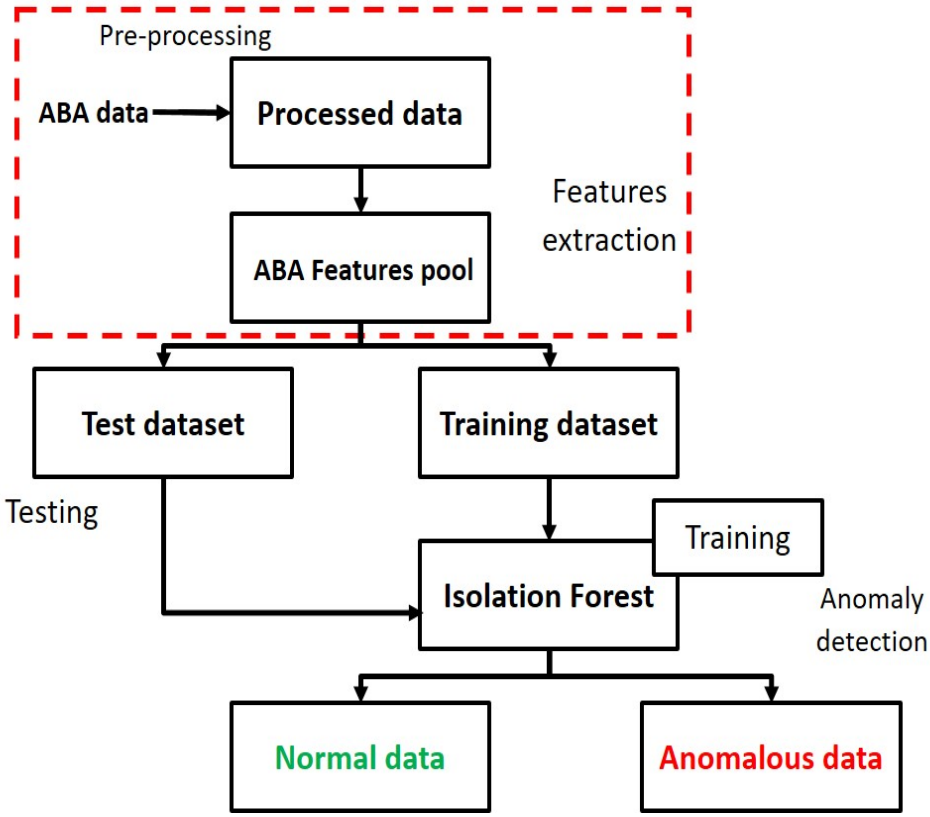


FIGURE 3.1: ABA based ML pipeline for rail condition monitoring system

3.2 Data Description and Acquisition

The required datasets for rail CMS exist in a disintegrated structure that requires pre-processing to bring it to the format that can be easily used as input to machine learning algorithm. Moreover, the pre-processed data gives a better representation of the condition of the system. The various data types acquired during data measurement campaign are stored in different databases. The ABA is one main dataset from these data types which is used for rail condition monitoring. However, the other data types that contains essential information, also need to be processed to prepare the final dataset. A dedicated train is used for data acquisition that has various sensors installed on it. The train ABA is captured using accelerometers in the measurement train while location information and rail images are captured by GPS and camera respectively. The data acquisition setup, sensors positioning on measurement trains and dataset structure is explained here.

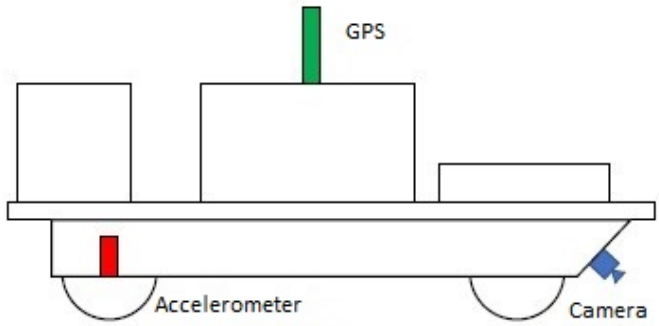


FIGURE 3.2: The sensors' arrangement on the inspection coach: video cameras (blue), GPS antenna (green), and the accelerometers (red)

The data capturing devices, installed on the measurement train are (i) a global positioning systems (GPS) for location information, (ii) six video cameras each side for shooting the rail and (iii) tri-axial accelerometers attached to axle-box on each

both sides of the train. The arrangement of the equipment is illustrated in figure 3.2. All these captured information are essential for the system, accelerometers data is used for anomaly detection, GPS data for locating the anomalies on track and video data is harnessed for validation of the detected anomalies. The accelerometers and GPS sensors are installed on the vehicle with a 2.4 m horizontal gap from each other while the GPS sensor is 7.5 m away from the video camera, see figure 3.2. Due to this structure of sensors placement, the data collected from these sensors must be synchronized in time and space to correspond to the same position of the rail.

The measurement train operates in either pushing (locomotive at the back) or pulling mode (locomotive at the front) during data acquisition. Therefore, either the accelerometers or the video camera will arrive first when running over a particular rail section. So the moving direction of the vehicle must be known. Knowledge on the train direction is critical for validation of the anomalies. It enables the system to compare the corresponding rail image to the ABA data. For synchronizing the acquired data, two types of counters are used: an external counter which increments each 1 mm and an internal counter that increases after each 0.25 mm. When the ABA system comes first at a certain point on the rail, the video camera captures the same point after 3150 external counters (distance of 3.15 m) are passed and vice versa. The sampling frequency of the ABA system is kept 25.6 kHz. The sampling rate of ABA is based on time (fixed frequency) while the counters have a distance based sampling rate that depends on train speed.

3.3 Data Structures

3.3.1 ABA Data

The raw form of the ABA data is saved as *.tdms format, details of which are given in the research article (NI, 2019). Furthermore, the acquired accelerometer data has been converted into time based and distance sampled data, which is saved in HDF5 files with naming convention as *.time.h5 and *.dist.h5 respectively. The *.time.h5 ABA data is sampled at the sampling frequency of the system, while in *.dist.h5 file 1mm distance is passed with each external counter. Both the time and distance based ABA data have similar structures as shown and explained in Table 3.1. The only difference in the structure of both these data files is that the time sampled files are indexed with increasing integers while the distance sampled files are indexed by internal counters. The ABA database contains data from the channels A and B. Data for each channel consist of 3 columns which show axle-box acceleration in *X*, *Y*, *Z* axes respectively. Each data point has its own internal counter which is unique throughout the dataset.

3.3.2 Video Data

The measurement train also captures the rail images by using high definition cameras. Data acquired by video cameras are saved in *.vdo files associated by their configuration files: *.event.txt and TrackNetCfg.s3db. There are multiple cameras installed on the vehicle, each of which is given an ID. The cameras that capture rail video on the right side of the side have IDs: 60, 61, 62, 63, 64 and 65. IDs of the cameras on the left side are: 70, 71, 72, 73, 74 and 75. Images for a certain length of rail track can be extracted from the video using the corresponding external counters

TABLE 3.1: The attributes and description of time-based ABA database

Attributes	Description
Internal counters	Data counters related to each sample of ABA data, unique for all ABA data for the same track on the same day
CHA1	ABA data on X-axis of the accelerometer A installed on left side of the train
CHA2	ABA data on Y-axis of the accelerometer A installed on left side of the train
CHA3	ABA data on Z-axis of the accelerometer A installed on left side of the train
CHB1	ABA data on X-axis of the accelerometer B installed on the right side of the train
CHB2	ABA data on Y-axis of the accelerometer B installed on the right side of the train
CHB3	ABA data on Z-axis of the accelerometer B installed on the right side of the train

associated with that location. The video data from cameras with ID 61 and 71 are used for validating the anomalies that will be explained in chapter 5 of the thesis.

3.3.3 GPS Data

The geographic location data associated with ABA are saved in *poi.csv files in the database that provides the route information. The GPS data is indexed with external counters which can be used to sync ABA data with their geographic location. This data is captured each 5 m of rail track. The GPS data is highly important to locate the anomalies on the rail track and to report it for maintenance.

3.3.4 Auxillary Data

The ABA data require other information during the pre-processing phase. The seg.csv files contains information about the direction of the inspection train (ERS-DIR). This value indicates whether the accelerometers come first or the video camera during data acquisition. It also provides the information about the segment of the rail track that the measurement train is inspecting such as SPOORTAK, GEOCODE, and GEBIED. Moreover it contains the route information of the inspection train (KM-FROM and KM-TO).

3.4 Data Pre-processing

As mentioned earlier, ABA data requires pre-processing in order to prepare it for ML technique and further analysis. It is the entry point of the machine learning pipeline for anomaly detection, shown in figure 3.1. In this chapter, only the first two blocks from the schematic diagram, pre-processing and feature extraction will be discussed. The steps involved in the data pre-processing are (i) Data filtering and synchronization (ii) Data calibration and (iii) The channel puzzle.

3.4.1 Data Filtering and Synchronization

As mentioned above, the measurement train collects three types of data, i.e. accelerometers data, GPS data and video data. These data types are stored in different databases. A counter is used to synchronize various datasets so that anomalies in ABA can be given a correct position and can be compared with the corresponding rail images during validation and performance metrics calculation.

The counters for synchronization initiate after some time the vehicle starts rail inspection, therefore not all the data in ABA database can be synchronized. Hence,

TABLE 3.2: Attributes and description of Sync database

Attributes	Description
Internal counter	Value associated with each ABA data point, increments every 0.25 mm
External counter	Value coupled with ABA data instances, ticks every 1 mm
Synchronization	Provides the starting point where the internal and external counters are synchronized
Time	Timestamp assigned to ABA data

it is important to use only that data where information about the synchronization is available. The sync.csv files provide the initial internal counter where the synchronization is started. Using this initial internal counter, all ABA data that precede this counter is filtered out. To synchronize the remaining data, an interpolation is calculated using the builtin Python interpolation function from the known variables in the sync.csv file i.e., internal and external counters:
$get\_extcount$ = **interpol**(int_{count}, ext_{count}, kind='$linear$'). The external counters for ABA are determined by feeding the internal counters to the function:
ext_{count} = **get_extcount**(int_{count}).

3.4.2 Data Calibration

Data calibration is essential if ABA data from both the channels are given as input to the anomaly detection model simultaneously. Two tri-axial accelerometers are used to capture the acceleration of the axle box of the train. Sensors on both sides of the measurement train are not aligned by default. Therefore data for one channel need to be rotated in order to bring a conformity in both the datasets. Figure 3.3 illustrates two unaligned tri-axial sensors in their X and Z axes. The issue with rotation of data from a channel is that the angle of rotation is not known. The information about the misalignment of the sensors has not been noticed during

data acquisition. The data is rotated with a random angle of rotation initially and compared with the reference data. This process is continued until the best match of the datasets from both the channels is obtained. So the rotation is entirely based on trial and error. This is a bit time consuming and more importantly unreliable. Therefore an alternative approach is used to deal with this problem. A transformed value P is calculated by taking the square root of the squared sum of the X and Z axes. This approach takes the direction out of equation and considers only the magnitude of the acceleration. It is not claimed to be the ideal solution but better than considering acceleration in unaligned X and Z directions separately.

$$P = \sqrt{x^2 + z^2} \tag{3.1}$$

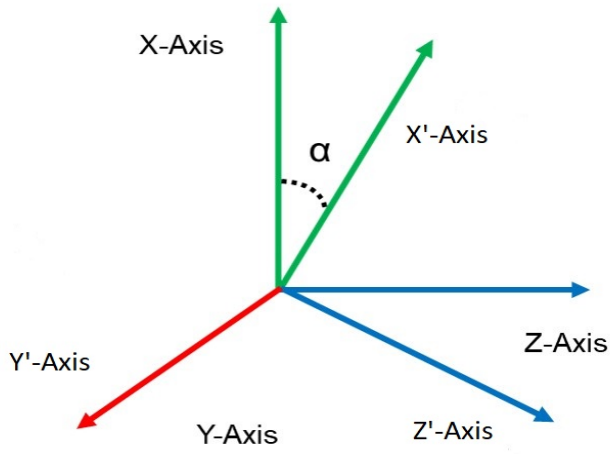


FIGURE 3.3: Tri-axial accelerometer coordinates for channel A, B

TABLE 3.3: Channel puzzle

Track-Dir	ERS-Dir	CH-A	CH-B	Right	Left
OP	1	Right	Left	CH-A	CH-B
OP	-1	Left	Right	CH-B	CH-A
AF	1	Left	Right	CH-B	CH-A
AF	-1	Right	Left	CH-A	CH-B

3.4.3 Channel Puzzle

Channel *A* and *B* in the ABA dataset do not always point to the same side of the rail. It depends on the values of both *Track-Dir* and *ERS-Dir* that tells about the direction and operating mode of the train respectively. Using these information, the ABA for left and right side track can be identified. The datasets do not explicitly provide these information. The measurement train collects data either in pushing mode in which the vehicle pushes the carriage, represented by a certain value of *ERS-Dir* in the dataset, or in pulling mode which is the other way around. Identifying data for pushing and pulling mode of measurement train is vital because it is found that these mode of operation have an impact on the ABA. The ABA data for the same track but in different modes have different patterns.

Therefore, the factor of pushing and pulling mode of the train need to be considered during processing and anomaly detection. During anomaly validation, it is important to know which side of the track the data is coming from, so that the corresponding rail images are used correctly. As mentioned above, there is a gap in the placement of video camera and accelerometers on the inspection train. Therefore, during validation process, the ABA need to be adjusted to images by adding or subtracting the external counters depends on whether the train is in pushing mode or in pulling mode. The issues regarding channels, train direction and operating mode are solved using the puzzle given in table 3.3.

3.5 Features Engineering

The data has been pre-processed prior to the time domain analysis for features extraction which made it well-structured, organized and more informative. At this point, the ABA data is still in its original raw form in which it is acquired. The sensory data in its raw form do not often reveal meaningful insights about the condition of the system. Therefore some features need to be extracted from raw ABA data that provide a better representation of the condition of the system. The process of feature extraction plays a vital role in machine learning based problems i.e. anomaly detection, object classification, and forecasting etc.

The benefits of feature extraction are two-folds. Firstly, it reduces the massive size of dataset by down sampling during feature extraction. Secondly, these features are more useful and clearer to detect anomalies or patterns of interest which helps ML model to learn faster and better. Extraction of signal features for monitoring the condition of a system is highly effective as these features can better reflect the normal and abnormal condition of the system (Assis Boldt et al., 2015; Islam, Khan, and Kim, 2015). To pull out the maximum possible insights from a signal regarding the health of any system, various features extraction paradigms have been used in the literature. Features are usually calculated using time domain, frequency domain and time-frequency domain analysis that makes a heterogeneous feature pool. However, this work uses the features based on time-domain analysis.

3.5.1 Sliding Window Approach

Time domain signal features are extracted from the raw time based ABA signal by applying a sliding window approach. A certain size of window is chosen that slides over the entire dataset with or without replacement. The approach is illustrated in

figure 3.4. Number of features are extracted for each individual window while sliding through the entire dataset. The size of the sliding window has a high impact on the final outcome of the system. The choice of size of sliding window also depends on how much of a track length needs to be checked for anomalies that suits the stakeholder's requirements. Anomalies with a precision of 1 to 2 m track lengths are considered as acceptable for this use case.

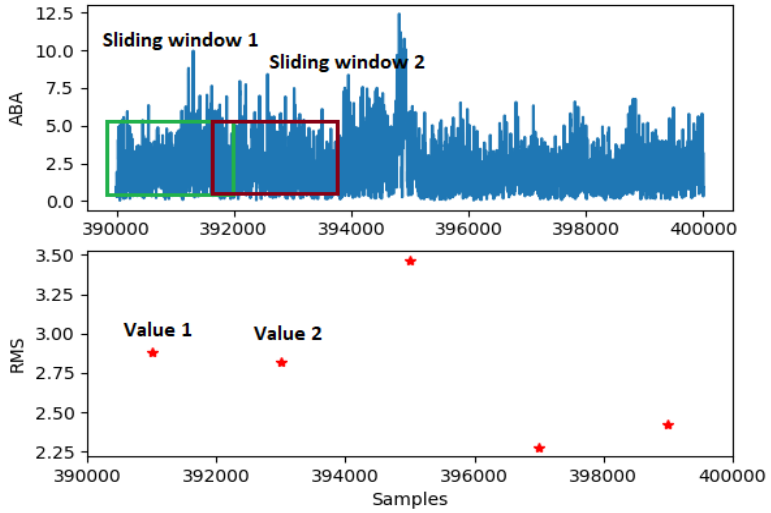


FIGURE 3.4: Illustration of sliding window for feature extraction.

The parameters, i.e. size of window, ratio of replacement of sliding window, can be tuned iteratively and the most optimal values should be selected that provides best performance and meets the stakeholder's needs. A sliding window of size 2000 is used during feature extraction, which represent the accelerometer data for approximately half a meter. The length of track covered by a sliding window depends on the train speed as well which in this case is considered as constant or ignored by the model. The extracted feature from each window of data samples

is assigned the mean value of the internal counters for that specific window. The window slides over the dataset with a 25% overlap. The overlap is important and done in order to reconsider the broken pattern at the end part of the signal from the previous window.

3.5.2 ABA Features Pool

A number of statistical features are extracted from the ABA data by applying the sliding window approach using time-domain analysis. The obtained features include root mean square (RMS), kurtosis value (KV), skewness value (SV), peak-to-peak value (PPV), crest factor (CF), and impulse factor (IF). The peak-to-peak feature with its raw ABA data is illustrated in figure 3.5. An extensive feature comparison and performance analysis is required to find out the optimal set of features in train ABA data because it is an unsupervised problem. The mathematical formulae and description of these statistical features are given as follows:

- **Root mean square (RMS)**: In mathematics, the RMS is defined as the square root of the mean square. It is also known as the quadratic mean and is a particular case of the generalized mean with exponent 2.

$$RMS = \left[\frac{1}{N} \sum_{i=1}^{N} {x_i}^2 \right]^{\frac{1}{2}} \tag{3.2}$$

- **Skewness value (SV)**: In probability theory and statistics, skewness is a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean. The skewness value can be positive or negative, or undefined. It describes the shape of the probability distribution of data.

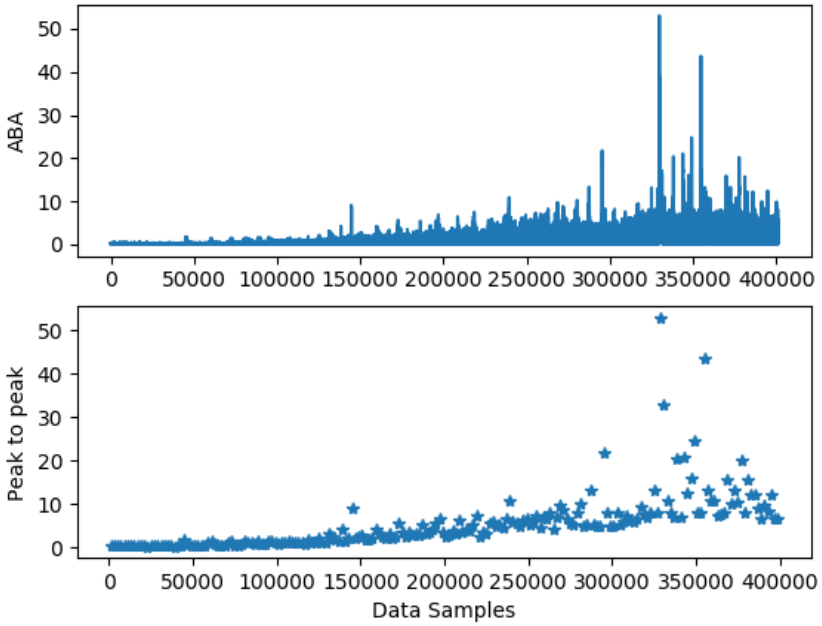


FIGURE 3.5: A raw ABA and its peak-to-peak feature

$$SV = \left[\frac{1}{N} \sum_{i=1}^{N} \frac{x_i - \bar{x}}{\sigma} \right]^3 \tag{3.3}$$

- **Kurtosis value (KV)**: In probability theory and statistics, kurtosis is a measure of the "tailedness" of the probability distribution of a real-valued random variable. In a similar way to the concept of skewness, kurtosis is a descriptor of the shape of a probability distribution.

$$KV = \left[\frac{1}{N} \sum_{i=1}^{N} \frac{x_i - \bar{x}}{\sigma} \right]^4 \tag{3.4}$$

- **Peak-to-Peak value (PPV)**: Peak-to-peak is the difference between the maximum positive and the maximum negative amplitudes of a signal.

$$PPV = max(x_i) - min(x_i) \tag{3.5}$$

- **Crest factor (CF)**: Crest factor is the peak amplitude of the waveform divided by the RMS value of the signal. In other words, crest factor indicates how extreme the peaks are in a signal.

$$CF = \frac{max(|x_i|)}{\left[\frac{1}{N}\sum_{i=1}^{N} {x_i}^2\right]^{\frac{1}{2}}} \tag{3.6}$$

- **Impulse factor (IF)**: In signal processing, the impulse factor is a ratio of maximum absolute value of signal and the mean of absolute value.

$$CF = \frac{max(|x_i|)}{\frac{1}{N}\sum_{i=1}^{N} |x_i|} \tag{3.7}$$

3.6 Conclusion

The ABA data is passed through a pre-processing step in which the data filtering, synchronization, calibration, and channel adjustment is performed. A pool of timed-domain statistical features are extracted from the pre-processed ABA data using a sliding window. The sliding window used during feature extraction, plays an important role in anomaly detection as well as in validation. The extracted features provide a better representation of the rail condition than the original raw data. The ABA features help the ML model in identifying the anomalies in ABA. Each feature has its own characteristic, for example, the kurtosis feature performs

well in identifying the early stage defects. Finding the best feature or best combination of features is highly important but hard to achieve in this case, because of the unavailability of actual outputs in the data.

Chapter 4

Anomaly Detection in ABA Data

4.1 Introduction

The designed machine learning (ML) pipeline for rail condition monitoring project consists of three main steps, (i) pre-processing (ii) feature extraction and (iii) anomaly detection. The first two steps are covered in chapter 3 of the thesis report. This chapter focuses on the anomaly detection part of the ML pipeline, which is the core task of the project. Various anomaly detection methods can be found in literature that uses different approaches to determine outliers in the data, i.e., statistical methods, classification-model based methods, density based approaches. Most model-based anomaly detection approaches, construct a profile of normal data points, and based on knowledge about the normal data, it can distinguish between normal and abnormal samples. Popular algorithms like classification-based methods (Abe, Zadrozny, and Langford, 2006), and clustering-based methods (He, Xu, and Deng, 2003), statistical methods (Rousseeuw and Driessen, 1999), all use this general approach. This profiling based approach has a couple of drawbacks: firstly, the model is trained to learn normal instances, but it is not optimized to detect anomalies. As a results, the detection accuracy of these algorithms may not be

as good as anticipated, causing too many false alarms or too many false negatives (the case in which an anomaly is considered as normal); secondly, most of the existing techniques work well for a low-dimensional and small size data but not good for data having high dimension and a massive size due to high computational complexity. The normal data profiling based approaches are not applicable in this use case because no prior knowledge about normal data is available.

From literature, various techniques for anomaly detection, among which one-class support vector machine (SVM), isolation forest (iForest), robust covariance, local outlier factor (LOF) were explored. None of these techniques is ideal for solving each problem as every technique has its advantages and disadvantages. The challenge is to find the right technique that provides a befitting solution to the problem. The above mentioned anomaly detection techniques were trained and tested on a synthetic dataset, a dataset which has true outputs and can be compared with predicted outputs for performance analysis of these algorithms, which is not the case with ABA data. Based on the performance yielded by these techniques and research recommendation, isolation forest is selected for detection of anomalies in the train ABA data. This chapter provides all the details about the anomaly detection using the Isolation forest algorithm.

4.2 Anomaly Detection

In ML problems, an unsupervised approach is used initially as a seed to generate labelled data unless the risk rules can be formulated based on domain knowledge for the problem. For some problems defining risk rules are easy, such as anomalies identification in network traffic metrics where the time between logins and

distance between origins can be used to formulate a risk rule. However formulating risk rules for identifying the probability of an employee committing securities fraud, is difficult. Here the behavioral data that the organization captures is very high dimensional and the relationship between the data attributes is complex. Hence without in-depth domain knowledge, formulating risk rules is difficult. Similarly in case of ABA based rail condition monitoring, there is no information available about the signal amplitude and frequencies in response to any defect on rail surface. Hence no definite risk rules can be formulated to reveal the relation between train ABA and rail defects. This combined with issues such as confidentiality makes it very hard to formulate and validate these risk rules. This is where the unsupervised ML techniques stand out to make the most out of the unlabeled data.

With very little domain knowledge, a simple unsupervised algorithm can be used to create a list of anomalies which can then be analyzed further to create labeled data. Once a sufficient amount of labeled data is generated by performing labelling task over a period of time, the paradigm of the ML technique can be transformed from an unsupervised approach to a supervised ML technique. This section specifically explains how outliers in the data are detected. The unsupervised anomaly detection is also referred to as outliers detection. In the context of outlier detection, the outliers/anomalies cannot form a dense cluster as the anomaly estimators assume that the anomalies are located in low density space. The ML pipeline shown in figure 4.1 depicts the implementation of the anomaly detection model enclosed in the rectangle.

The anomaly detection technique used in this project, which is known as Isolation Forest, is quite unique in its approach to detect outliers. It is a model-based method that explicitly isolates anomalies rather than normal data profiling. It has

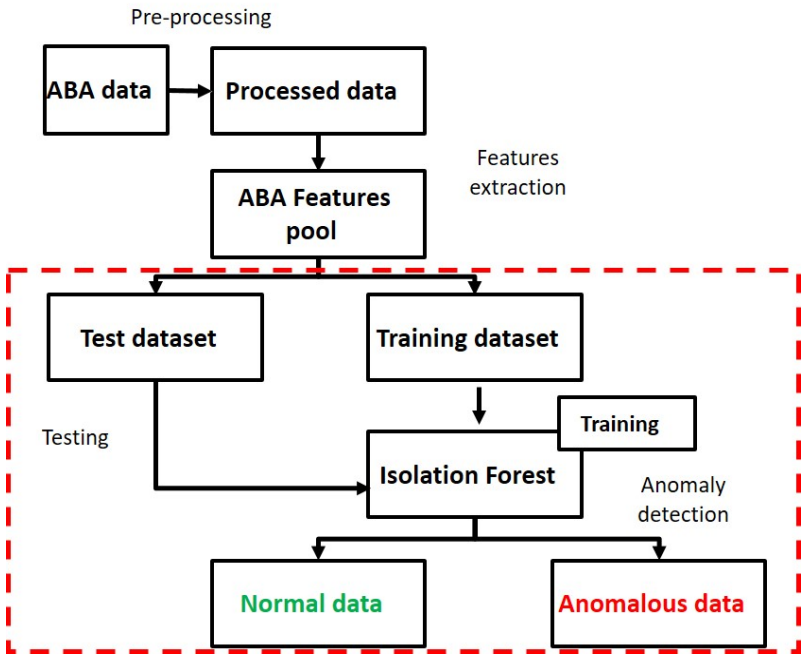


FIGURE 4.1: Machine learning pipeline for anomaly detection in ABA data

a linear time complexity with a low memory requirement. Literature reveals that iForest yields better performance compared to ORCA (a tool that uses nearest neighbor based approach), local outliers factor (LOF) and Random Forests algorithms in terms of area under the curve (AUC), and processing time especially in large data sets (Liu, Ting, and Zhou, 2008). The iForest algorithm achieves better results in high dimensional problems having a large number of irrelevant features, and also in situations where training data is purely normal. This technique works on the basis of two quantitative properties of anomalies: firstly, they are in minority, containing fewer data samples and secondly, they have attribute-values that are very different from those of normal instances. The algorithm perceives anomalies as "few and different", which make these instances easy to be isolated from normal data.

The isolation forest method builds an ensemble of trees called "iTrees", for a given data set, the data samples with a shorter average path length are considered as outliers by the algorithm. Two variable parameters are involved in this method: firstly, the number of trees to build and secondly, the sub-sampling size. It is reported that iForest anomaly detection performance converges quickly having a lower number of iTrees, and it only requires a small sub-sampling size to achieve high detection performance with high efficiency (Liu, Ting, and Zhou, 2008). The salient features of iForest that distinguish it from rest of the anomaly detection algorithms are:

- The isolation characteristic of iTrees enables them to build partial models and exploit sub-sampling to an extent that is not feasible in existing methods. Since a large part of an iTree that isolates normal points is not needed for anomaly detection; it does not need to be constructed. A small sample size produces better iTrees because the swamping and masking effects are reduced.
- Isolation forest does not apply distance or density calculations to find anomalies. This approach eliminates the high computational cost of distance calculation in all distance-based methods and density-based methods.
- This technique has a linear time complexity with a low constant and a low memory requirement.
- Isolation forest is capable of handling a massive size dataset with a large number of irrelevant features.

Isolation and Isolation Tree: The term isolation refers to "separating a data sample from the rest of the data". Outliers in data, are more susceptible to isolation because they are few in number and different from the dense data clusters.

Splitting of a feature is recursively repeated in a random tree until all instances are isolated. This random partitioning yields shorter paths for anomalies because of its distinguishable feature-values. Hence, when a forest of random trees collectively produce shorter path lengths for a certain data point, then it is highly likely that the data point is an anomaly. The number of splits required to separate an instance is equivalent to the path length from the root node to a terminating node in a tree. Figure 4.2 illustrates the concept of anomalies being more susceptible to isolation during random partitioning. It can be noticed that for a normal data point, x_i, it generally requires more splits in data to be isolated, while for anomalous data instance, x_o, the opposite is true; it usually requires fewer partitions to be separated from rest of the data. Hence anomalies have shorter path lengths. In isolation forest, partitions are generated by randomly selecting a feature i.e. kurtosis, peak-to-peak etc., and then randomly choosing split points between minimum and maximum value of the selected feature. The splitting of an attribute is performed recursively which can be represented by a tree structure.

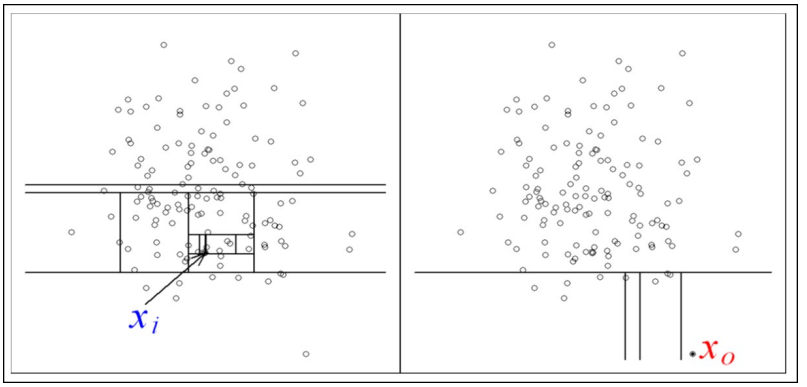


FIGURE 4.2: Normal point x_i requires more random partitions to be isolated and anomaly x_o requires fewer partitions to be isolated ((Liu, Ting, and Zhou, 2008))

Several key terms such as isolation tree, path length and anomaly score need to be defined in order to clearly understand the isolation forest algorithm:

4.2.1 Isolation Tree

Let T be a node of iTree, which is either an external node with no child or internal node with one test and exactly two daughter nodes (T_l, T_r). A test contains two parameters q and a split p such that the test $q < p$ divides data points into T_l, T_r. Figure 4.3 illustrates the structure of a binary tree.

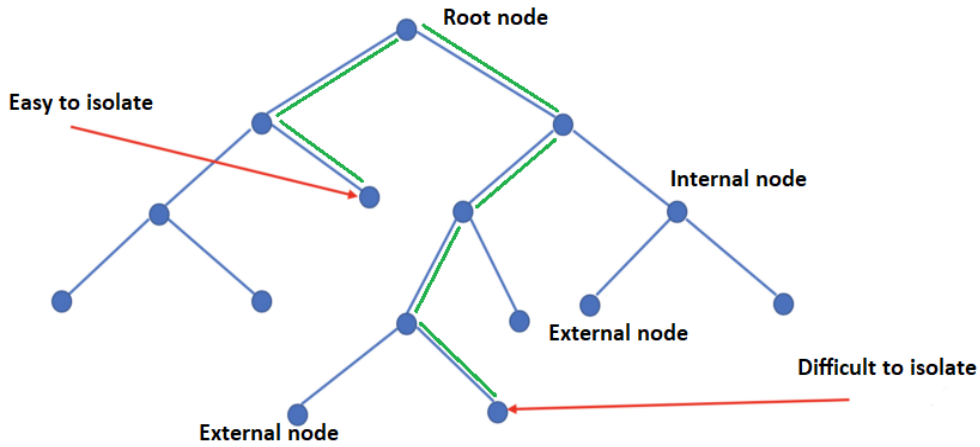


FIGURE 4.3: Illustration of a binary isolation tree

Assume a dataset $X = \{x_1, ..., x_n\}$ containing n number of samples with a d-variate distribution. To build an iTree, dataset X is iteratively divided by selecting a feature q and a split value p, until any of these conditions are satisfied: (i) the tree reaches a height limit, (ii) |X| = 1 or (iii) all the data samples in X have the same values. Isolation tree is like a normal binary tree and each of its node has exactly zero or two child nodes. If all the instances of dataset are distinct, each of it is isolated to an external node once the tree is fully grown. In this case the number of external

nodes is n while internal nodes are n - 1; adding these two parameters gives the total number of nodes in an iTree, which is $2n$ - 1; therefore the memory requirement is bounded and linearly grows with n.

4.2.2 Path length

The path length is denoted by *h(x)* and is defined as the number of edges an instance x_i traverses in an iTree from the root node until the traversal ends at an external node. The outliers generally have a shorter path length compare to inliers in the data. To illustrate this, a dataset containing normal and fraudulent credit card entries, is used. It is obtained from an online machine learning competition forum. The purpose of using this dataset is to demonstrate the calculation of path lengths by the iForest model for normal and anomalous data points. The reason why this dataset has been used instead of ABA dataset because it provides labeled data. Using labeled data, the path lengths for both normal and abnormal data samples can be calculated.

Figure 4.4 shows a histogram to illustrate the average path lengths for normal and anomalous data instances. The path lengths in this example are calculated using 15 trees with a sampling size of 5000. Each tree in the forest is generated with different set of data partitions. Therefore average path lengths are calculated over a number of trees to determine the expected path length. For anomalous data points, the shorter path lengths appear most of the times while for normal data instances the longer path lengths are yielded with high frequency.

4.2.3 Anomaly Score

The anomaly score for data instances is calculated on the basis of their path lengths. It is anomaly score which defines a data point as anomaly. The maximum possible

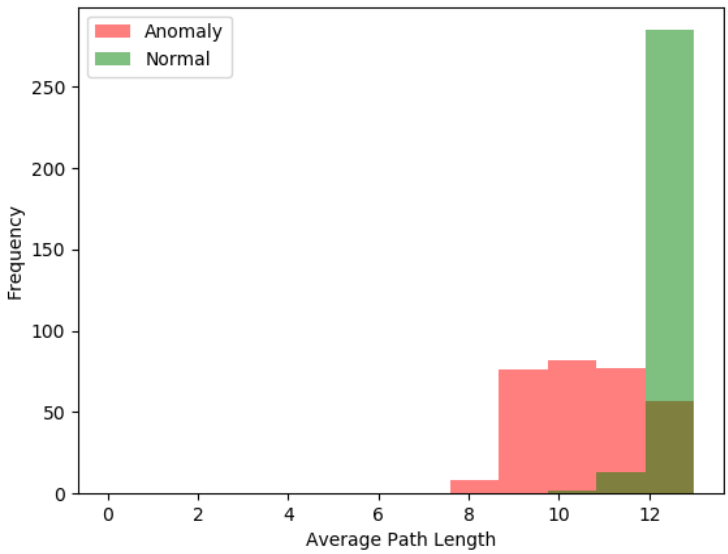


FIGURE 4.4: Path lengths comparison of path lengths for normal and abnormal data determined by iForest algorithm

height of an isolation tree grows in the order of n, while the average height grows in the order of log n (Breunig et al., 2000) . If *h(x)* is normalized by any of the above the parameters, it is neither bounded nor be compared directly. An iTree has similar structure to a binary search tree (BST); the calculation of average *h(x)* for an external node terminations is the same as the unsuccessful search in BST. Estimation of the average path length of iTree is thus inferred from a BST analysis. Given a dataset of n data points, section 10.3.3 of (He, Xu, and Deng, 2003) provides the average path length of an unsuccessful search in BST as:

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n} \tag{4.1}$$

In equation 4.1, *H(i)* represents a harmonic number and it can be estimated by

$\ln(i) + 0.5772156649$ (Euler's constant). While *c(n)* which is the average of *h(x)* given *n*, is used to normalize *h(x)*. The anomaly score *s* of a data sample *x* is given as:

$$s(x, n) = 2^{\frac{-E(h(x))}{c(n)}} \tag{4.2}$$

In equation 4.2, *E(h(x))* shows the average value of *h(x)* from a collection of iTrees, considering this equation for anomaly score *s* calculation, the following statements can be made:

- When *E(h(x))* is equal to *c(n)*, anomaly score s is 0.5.
- When *E(h(x))* is equal to zero, anomaly score *s* is 1.
- When *E(h(x))* is equal to *n-1*, then anomaly score *s* is 0.

The anomaly score *s* is monotonic to *h(x)* and using the value of *s*, the following assessment can be made:

- If the value *s* of an instance is close to 1, then it is definitely an anomaly.
- If a data instance has *s* value much lesser than 0.5, then it is considered as normal.
- If anomaly score *s* for an instance is around 0.5, then the entire data sample has no distinct anomaly.

4.3 Training of Isolation Forest Model

In the training stage of the model, iTrees are constructed recursively by a random selection of a feature from training dataset until data points are isolated or a specific tree height is reached which results in a partial model. It must be mentioned here that the limit of tree height *l* is set automatically by using sub-sampling size

Algorithm 1: *iForest*(X, t, ψ)

Inputs: X - input data, t - number of trees, ψ - sub-sampling size
Output: a set of t *iTrees*

1. **Initialize** *Forest*
2. Set tree height limit $l = ceiling(log_2\psi)$
3. **for** $i = 1$ to t **do**
4. $X' \leftarrow sample(X, \psi)$
5. $Forest \leftarrow Forest \cup iTree(X', O, l)$
6. **end for**
7. **return** *Forest*

$\psi : l = ceiling(log_2\psi)$, which is nearly the average height of binary tree (Breunig et al., 2000). The notion of average height reveals the interesting insight to make a decision about anomalies. Data samples that have greater path length than tree height limit are definitely normal, the interest lies in those instances that have shorter path lengths than tree height because those instances are most likely anomalies. The pseudo code of iForest training is given in Algorithms ***1*** and ***2***.

Two parameters are fed to the iForest algorithm, the sub-sampling size ψ and the number of trees t. It is found that the sub-sampling size ψ =128 yields sufficient insight for anomaly detection. Moreover, the path length converges well before t=100. At the end of the training process, an ensemble of trees is returned which can be referred as the trained model. The algorithm uses the constructed trees during training to test new data for anomaly detection.

4.4 Model Evaluation and Discussion

In order to evaluate the trained model, an anomaly score s is determined from the expected path length $E(h(x))$ for each instance in test dataset. $E(h(x))$ is calculated

Algorithm 2: *iTree(X,e,l)*

Inputs: X - input data, e - current tree height, l - height limit
Output: an *iTree*

1. **if** $e \geq l$ *or* $|X| \leq 1$ **then**
2. **return** *exNode{Size* $\leftarrow |X|$ *}*
3. **else**
4. let Q be a list of features in X
5. randomly pick a feature q $\in Q$
6. randomly select a split point p from max and min values of feature q in X
7. $X_l \leftarrow filter(X, q \leq p)$
8. $X_r \leftarrow filter(X, q \geq p)$
9. **return** *inNode{Left* $\leftarrow$ *iTree(*$X_l, e+1, l$*), Right* $\leftarrow$ *iTree(*$X_r, e+1, l$*),*
SplitAttr $\leftarrow$ q, SplitValue $\leftarrow$ p *}*
10. **end if**

by traversing instances through each iTree in iForest. Using the *PathLength* function, given in Algorithm ***3***, a single path length *h(x)* is determined by counting the number of edges e from the root node to a terminating node as data instance x passes through an iTree. When x is terminated at an external node, where $Size > 1$, the return value is $e + c(Size)$. The adjustment in the form of *c(Size)* accounts for an unbuilt subtree beyond the limit of tree height. When *h(x)* is obtained for each tree in the collection of trees, an anomaly score is yielded by computing *s(x, ψ)* in equation 4.2. The complexity of the model evaluation process is *O(nt log ψ)*, where n is the size of testing dataset. To obtain the top m anomalies, the data is sorted in descending order using s. The first m instances are the top m anomalies depending on the value of s.

The training ABA data do not provide prior information about normal and abnormal data instances, in other words no ground truth information is available to validate the model. Which is why, an unsupervised technique has been used. The accuracy of the iForest model cannot be determined in the absence of ground truth

Algorithm 3: ***PathLength(x,T,e)***

Inputs: X - data instance, T - an iTree, e - current path length; to be initialized as 0 at the function first call
Output: path length of x

1. **if** T is an external node **then**
2. **return** $e + c(T.size)$ {c(.) is defined in 4.1}
3. **end if**
4. a ← *T.splitAtt*
5. **if** $x_a <$ *T.splitValue* **then**
6. **return** *PathLength(x, T.left, e + 1)*
7. **else** $\{x_a \geq T.splitValue\}$
8. **return** *PathLength(x, T.right, e + 1)*
9. **end if**

information about anomalies. However, visualizing the results in 2D plots indicate to some extent the performance of the model in separating anomalies from normal data. A 2D contour plot is given in figure 4.5, which shows the detected outliers in red and yellow for test and training dataset respectively, while the normal data points are colored in green and white for test and training dataset respectively. It can be noticed that data instances which fall beyond the learned threshold are declared as outliers by the model. Figure 4.5 illustrates the results for two time-domain statistical features from the extracted pool of features namely the kurtosis and peak-to-peak values. Various combination of features are used during testing phase of the model which will be discussed in chapter 5.

4.5 Conclusion

The isolation forest algorithm separates oultiers based on the assumption that the outliers are few and different from the dense cluster in the data. This algorithm

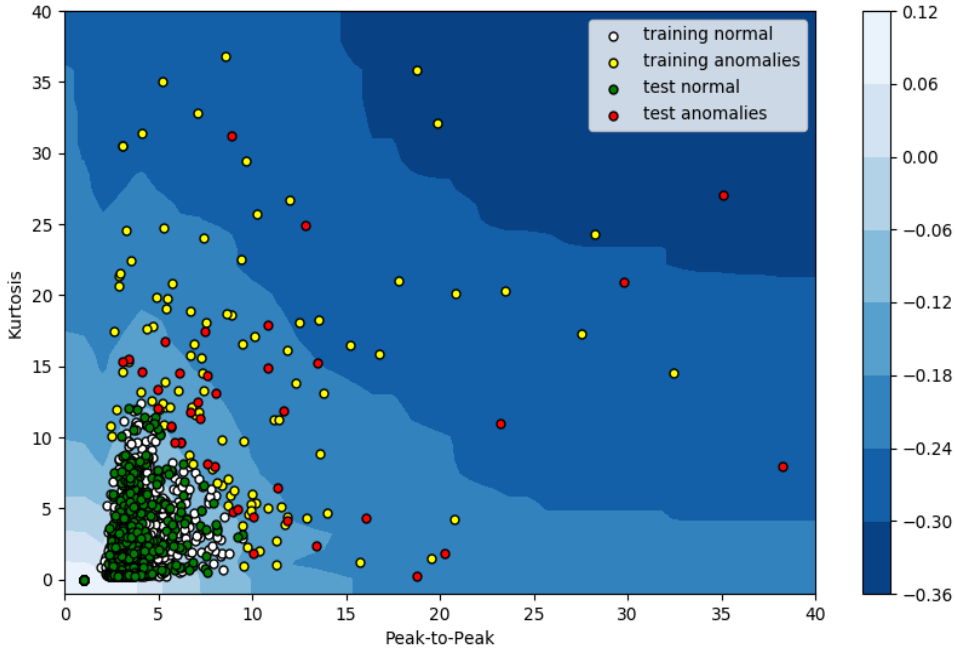


FIGURE 4.5: Illustration of iForest model to separate anomalies from normal data cluster.

identifies anomalies by looking into the disparity between anomalies and the normal data, unlike other model-based algorithms that profile normal data for anomaly detection. The isolation forest algorithm generates iTrees using a sub-sample of data that reduces the effect of swamping and masking. Besides that, the technique has a linear time and computation complexity and low memory requirement. Performance metrics of the iForest model can only be determined if true labels were available for ABA data. The parameters such as impurity ratio, sub-sampling size, number of iTrees, type and number of the features affect the outcome of the iForest algorithm.

Chapter 5

Validation and Analysis

5.1 Introduction

The anomaly detection model is used to find out abnormal patterns in train ABA data. For this purpose, the isolation forest algorithm is used which is discussed in detail in chapter 4. The sensitivity of the model for detecting anomalies can be altered by tuning its parameters that generates different outputs. Usually in a supervised machine learning approach, a validation dataset is used to validate the model. The applied anomaly detection method in this design problem works in an unsupervised manner as the true outputs are not available. This implies that the validation process here refers to the feasibility of usage of ABA for rail conditioning monitoring, not to the model accuracy. In this validation approach, the detected anomalies in ABA are compared with the corresponding rail images to see whether the predicted anomalies correspond to actual anomalies on the rail or not. Performance metrics such as true positives, false positives and false negatives etc. are calculated during this analysis. Various other approaches for assessing the feasibility of ABA will also be discussed in this chapter.

5.2 Validation Approaches

Validation and analysis of anomalies in ABA can be performed in several ways using the available auxiliary datasets i.e. rail images data and EC testing reports. Channel and passage comparison for anomalies is also conducted to find out the behavior of ABA for various passages of a train on both sides of the rail. Comparison of ABA anomalies with the respective rail images in section 5.2.1 helps to determine the performance metrics, while using the EC testing reports in section 5.2.2 helps in locating the head-check defects and provides their severity levels. That in turn provides an opportunity to explore the relation between magnitude of the ABA abnormality and the head-checks severity. Analysis of ABA data that is acquired for both channels i.e. left and right rail, in section 5.2.3 is done in order to study the effect of defects on data for both the channels. Finally another analysis to assess the applicability of ABA for rail condition monitoring is performed in section 5.2.4. The ABA for multiple train passages at a certain route either in same direction and or in opposite direction are processed. The interest lies in comparison of the anomalies from different passages and to check the consistency of ABA signal in response to rail abnormalities. These analyses serve as proof of concept for using train ABA for rail defect detection.

5.2.1 Validation using camera images

The ability of the anomaly detection model to distinguish between normal and abnormal data points, can be assessed by visualizing the results of the model in a 2D contour plot. The plot is shown in figure 4.5 of chapter 4. It can be seen that the algorithm accurately separates the outliers from normal ABA data points with a few

exceptions. In this way, the model is validated or in other words the anomaly detection potential of the model, is assessed. However, the performance cannot be quantified and performance metrics such as accuracy, hitrate and false alarms etc. cannot be calculated. Therefore, rail images of the same track are used for comparison of the actual defects on rail with the detected anomalies in ABA. The rail images based validation method is depicted in figure 5.1.

The following steps are taken in the validation process of ABA anomalies using rail images:

- Determine the external counters of the data samples that are reported as anomalies by the algorithm.
- Use the external counters of the detected anomalies to extract the corresponding camera images of the rail from the video.
- Manually label the defects such as head-checks and squats on the rail using the **labelImg** tool. It returns pixel values of the marked spots in the images.
- Process these images to spot the detected anomalies on the rail images using their external counters.
- Synchronize the external counters with the pixels of the defects on rail images. Execute the python script that compares the counters of anomalies and actual defects.
- Calculate the performance metrics for the anomaly detection model in terms of hit rate, false alarms and false negatives.

The performance metrics are determined by comparing the model output with defects labeled on rail images. When the actual rail defect and the detected ABA anomaly are found within a predefined range (approximately 1 m distance on the rail), the anomaly is regarded as a hit (true positives), otherwise it would be counted

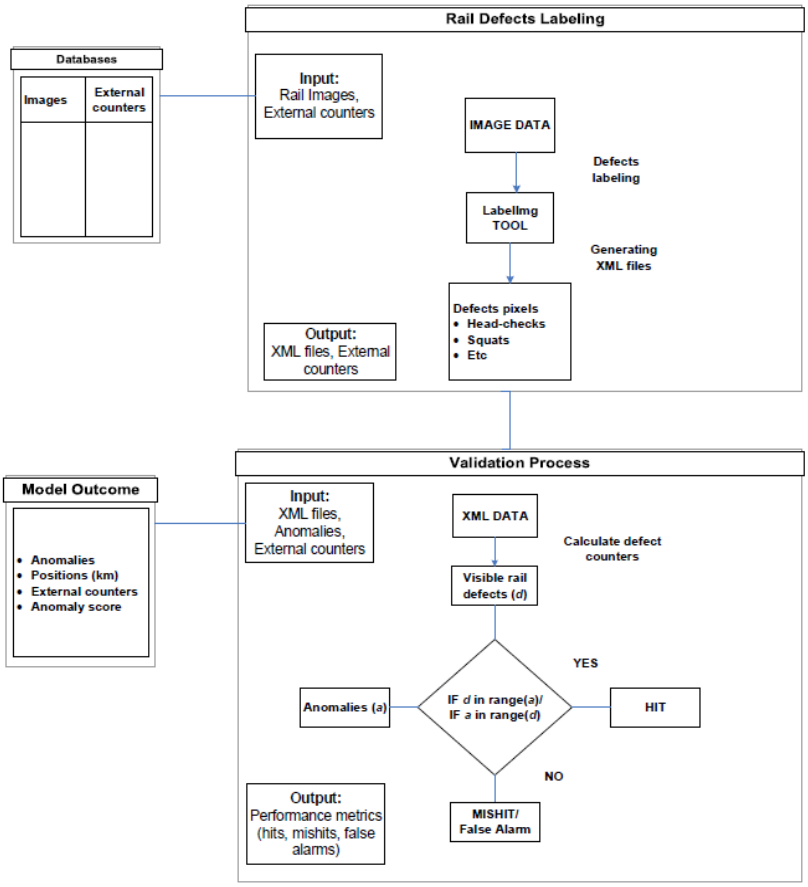
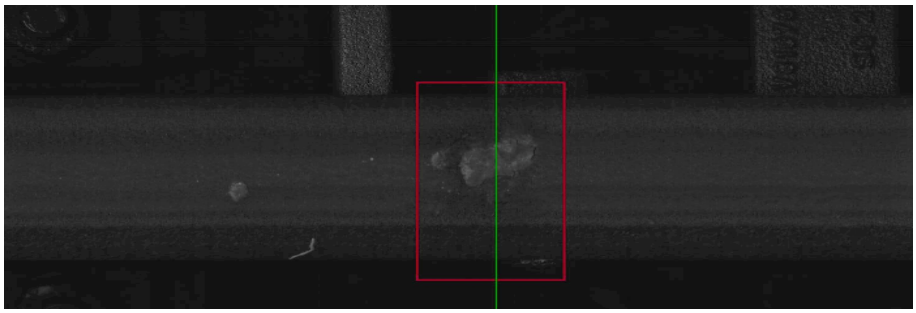
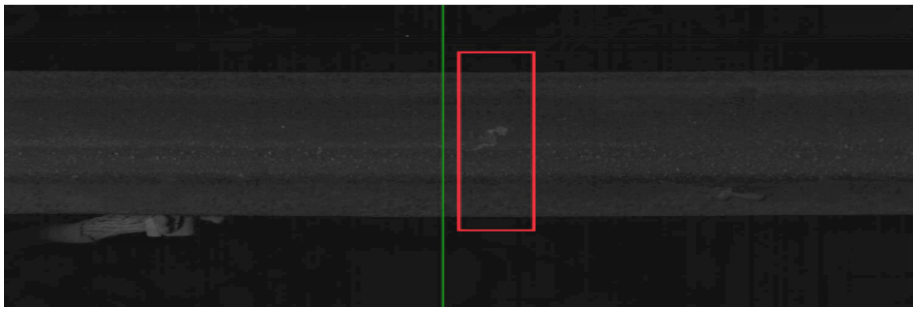


FIGURE 5.1: Validation process of the rail condition monitoring system

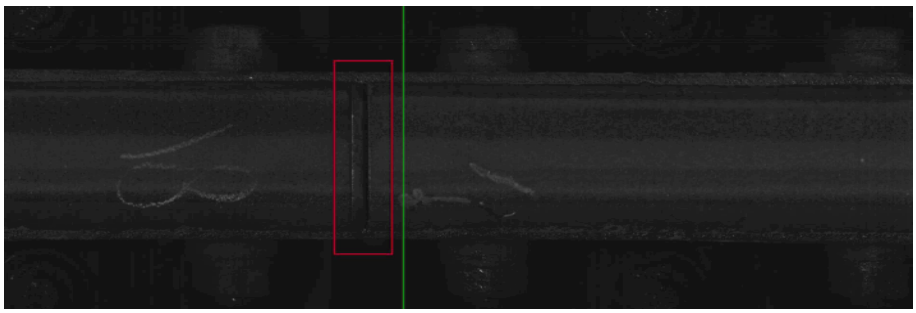
as a false alarm. Figure 5.2 shows three examples of a hit because the rail abnormality (enclosed in rectangle in red) and the predicted ABA anomaly (shown as vertical line in green), are within the specified range.



(A) Example 1



(B) Example 2



(C) Example 3

FIGURE 5.2: Comparison of anomalies on rail and ABA

5.2.2 Validation by ECT data

The aim of the ABA based condition monitoring system is to detect early stage rail surface detects. From the analysis on rail images, it is found that the model is capable of detecting the irregular ABA patterns for clearly visible rail surface abnormalities and the rail objects as shown in figure 5.2. Synchronizing and comparing these defects with anomalies in the data is easy because these rail abnormalities are visible in the images and therefore can be labeled. In case of head-checks, the labelling is difficult because head-checks are hardly visible on the rail images. Hence the anomaly comparison is uncertain. Here utilizing ECT reports come in handy because these reports have details about head-checks i.e. their location (km), and crack size. These reports provide head-checks location as a track section rather than a single point on rail. Moreover, the depths of the head-check cracks are given in ranges rather than exact depths. The positions of ABA based anomalies are determined in terms of kilometers so that it can be compared with that of ECT results. All the track sections where ECT has reported head-checks have anomalies in ABA. These results are quite promising as the ABA based anomaly detection model is capable of detecting head-check defects that develop into severe cracks as the train passages increases and ultimately cause rail failure. Besides detecting the head-checks, the ECT data based validation helps in performing head-checks severity analysis.

An attempt has been made to investigate the relation between size of the rail defects, head-checks in particular and level of abnormality of ABA signal. As mentioned earlier, the anomaly detection model detects anomalies at the reported locations of head-checks, allowing a comparison between the details of ABA anomalies and the head-checks. Information regarding anomalies i.e. location and ABA

anomaly score are generated by the rail CMS, while details of head-checks i.e. location and crack size are obtained from ECT reports. The hypothesis is that the ABA anomaly score and the corresponding defect crack size hold a highly correlated relation. If that is found to be true then based on information of the other, one can be estimated. The plot in figure 5.3 illustrates the relation between crack depth and anomaly severity for a rail track at the Almere-Weesp route. These anomalies are found at those locations where ECT has reported head-checks. The results obtained from this analysis are not true to the expectations and prior assumptions. The plot shows the corresponding head-check cracks size in green and anomalies score in red. Looking into the yielded outcome, the graphs follow each other for some anomalies, but for others the anomaly scores are just opposite to the crack sizes. Hence, no concrete conclusion can be drawn because of insufficient correlation.

5.2.3 Channel comparison

It is expected that severe defects on the rail create clear patterns in the ABA signal not only on the rail side where the defect exists but also in ABA data of the other side of the rail. Having this assumption in mind, the ABA data for both channels A and B are processed and synchronized. For each ten meters track section, the ABA signals with their anomalies, were plotted for both channels against each other, see figure 5.4. The anomalies are shown in red vertical lines exactly at the location where it is detected. It is found that in most cases anomalies appear on both channels at the same location while sometimes it is seen on one side only. Intensive research and analysis is required to understand this behavior of ABA. A possible explanation is that, early stage rail defects affect the ABA on just one side while severe defects or rail objects have influence on ABA data on both sides.

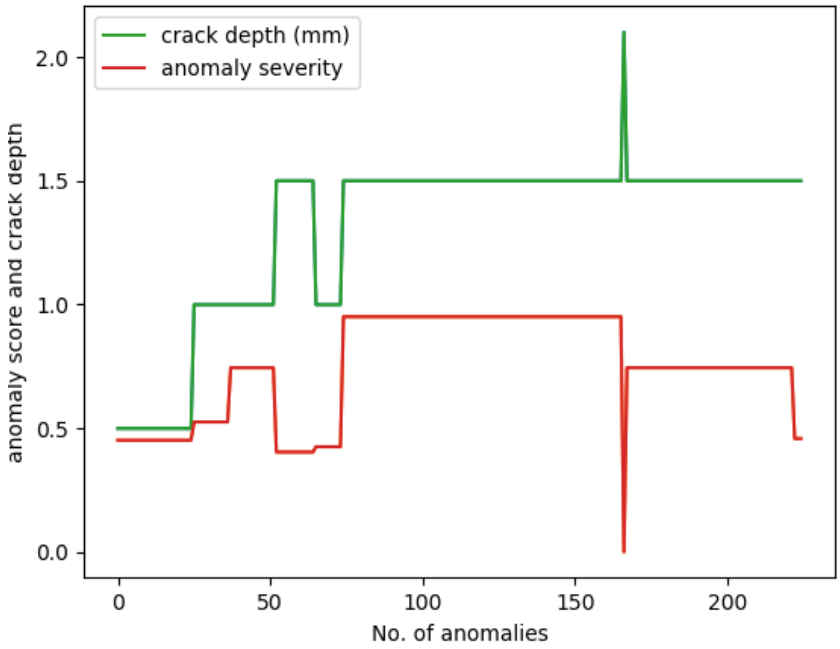
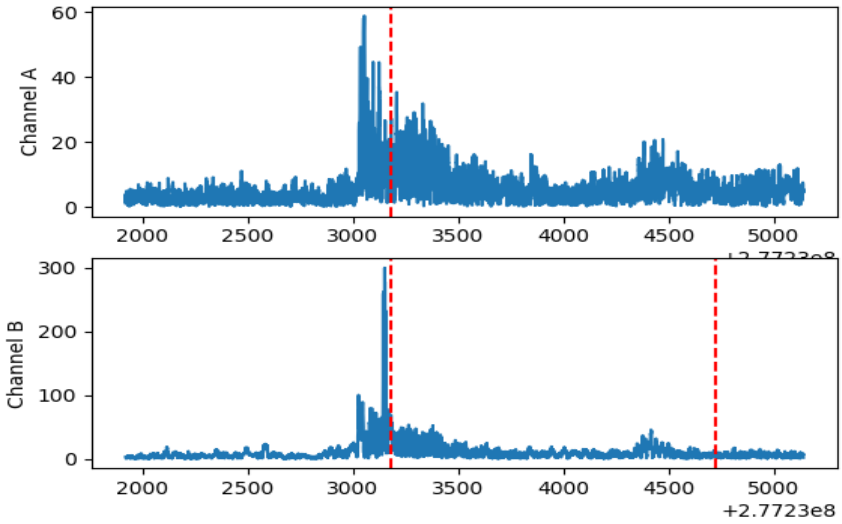


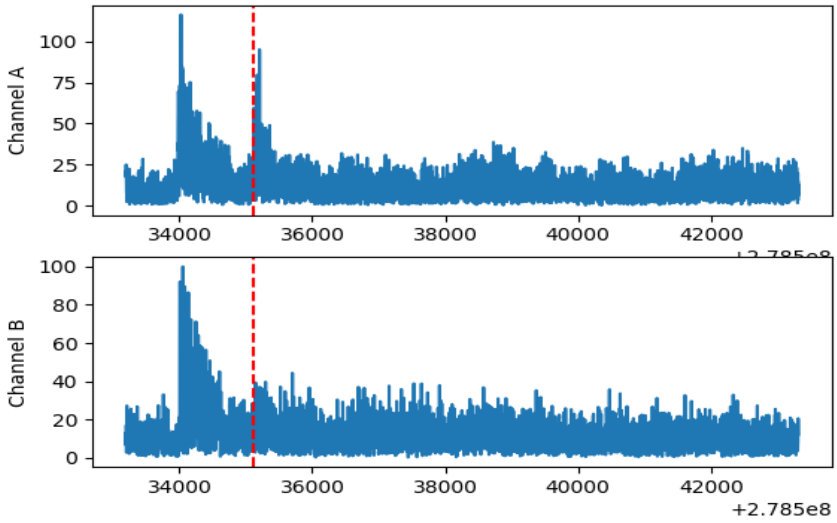
FIGURE 5.3: Severity analysis by comparing head-check (crack depth) with severity of ABA anomalies

5.2.4 Passage comparison

Passage comparison is used to check the consistency and feasibility of ABA. In order to do that, it is required to process the ABA data that is acquired from the same route for multiple train passages either in the same direction or in the opposite direction. Applying the anomaly detection technique on ABA data for each passage will reveal whether the detection of anomalies is consistent or not. Ideally, a high percentage of anomalies should be detected at the same location for each passage. In this analysis, the ABA data for two passages on the same route in Groningen are compared. In table 5.1 , pairs of anomalies that are colored with the same color represent the repetition of anomalies within a certain range for two different passages. The positions of the paired anomalies lie within approximately 1 m distance on the



(A) Example 1



(B) Example 2

FIGURE 5.4: Channel comparison by plotting the anomalies detected by the model on each side of the rail

TABLE 5.1: Repetition of anomalies at same location for two passages

Match	Location (km)	Counters
YES	7.079	299060431
	7.080	281822233
YES	7.081	308862381
	7.081	299062580
YES	7.082	281820014
	7.083	299064732
NO	7.085	299066877
YES	7.090	299069020
	7.091	281811139
YES	7.092	299073322
	7.093	281808923

rail track. External counters in each pair shows ABA measurements for different passages. It is found that the repetition of the anomalies at the same position on rail but different passages occur approximately 90% of the times, which shows the behavior of ABA is consistent. The consistency of the ABA supports its applicability for rail condition monitoring and defect detection.

5.3 Results and Discussion

It is reported in research literature that the kurtosis can be a good indicator to distinguish between a faulty and a normal system (Heng and Nor, 1998). When the system deteriorates, this value increases to indicate a failure. However, the kurtosis value decreases when the defect reaches an advanced state of degradation. Therefore, the kurtosis is most effective in detecting early stage defects, i.e. head-checks and squats in this case. The crest factor is the ratio of the PPV to the RMS and it is reportedly a good candidate for detecting incipient faults in system. However, this feature decreases with a progressive failure because the RMS value generally rises

TABLE 5.2: Outcome of the anomaly detection model for various ABA features

Features	Anomalies	Hits	False Alarms	Mishits
RMS	74	49	25	98
Kurtosis	85	70	15	77
Skewness	81	67	14	80
Peak-to-Peak	92	68	24	79
Crest factor	101	83	18	64
Impulse factor	93	75	18	72
All features	126	94	32	53

with a progressive failure. Thus, the crest factor could give better performance for detecting incipient defects while the RMS value can represent severe defects in a better way. The impulse factor is used to measure the strength of an impact generated by a defect in the system (Caesarendra and Tjahjowidodo, 2017). The skewness value measures the asymmetry of the impulse generated by defects.

The extracted pool of six features is tested by the iForest model for anomaly detection. The tests include the combination of all features as well as each feature individually as an input to the model. The performance yielded by these features is given in the table 5.2. The tests that were conducted using all features, yields better results compared to using individual features. Looking into the results produced by each individual feature, it can be noticed that the crest factor gives the best result. It achieves the highest number of hits with comparatively less mishits and false alarms. However, the test on the combination of all features outperforms each feature tested individually. In this analysis, the detected anomalies in ABA are regarded as hits if the anomaly lies within approximately 1 m range of the labelled rail defect. This range changes with variation in the sliding window, used during feature extraction. The false alarms are those anomalies which are reported by the model because these exist in the data but during validation nothing (rail defects or

objects) can be spotted at that location or its vicinity by looking into the rail images. This implies that the detected anomalies are false alarms just because the corresponding rail images do not show any rail defect or object. Mishits are the number of rail defects that go undetected by the model, which means that the defect or rail object exist at the location but no abnormality in the nearby ABA data is found.

A significant number of mishits and false alarms badly affects the reliability of ABA data to be used for condition monitoring of rail infra-structure. The reason for false alarms could be any other abnormality in the rail system besides rail defects i.e. rail misalignment, train wheel fault, surface disparity, measurement noise etc. Therefore, the false alarms can be expected in this case because of the mentioned factors. There is a fair amount of hit points (a case in which the rail defects and the anomalies coincide) which supports the feasibility of ABA, however the considerable number of mishits in the model results require further improvements. The results given in table 5.2, are obtained from a dataset of a single track in Groningen. Therefore a definite conclusion about the applied approach cannot be drawn by experimenting only one dataset.

5.4 Conclusion

The output of the ML pipeline is validated using the rail images. The performance metrics of the rail CMS are calculated on the basis of comparison between actual irregularities spotted on rail images and the detected anomalies in ABA. If the detected ABA anomaly lie within 1 m range of the actual rail defect, it is considered as hit otherwise a false alarm. Analyses on channel and passage comparison are performed in order to check the feasibility of ABA for rail condition monitoring. These

analyses reveal that the ABA has enough potential to represent the rail abnormalities.

Chapter 6

Graphical User Interface (GUI) Design

6.1 Layout design

The user interface is an essential component of any software based system to provide an interaction between the system and the users. A python based graphics library *PyQT* has been used to make the layout of the rail CMS. The design of the GUI is shown in figure 6.1 which consists of several widgets, buttons, and tables etc. The user interface provides a limited control of the rail CMS to the maintenance personnel according to the requirements defined by Strukton. The user can tune parameters such as sliding window size, type of features and sampling size etc., while performing pre-processing and anomaly detection by using the interface. A wrong input to the model will generate an error notification. In the notification, the correct format of the required file will be suggested. Moreover, the user will be notified with a run-time error in case a problem occurs during the process.

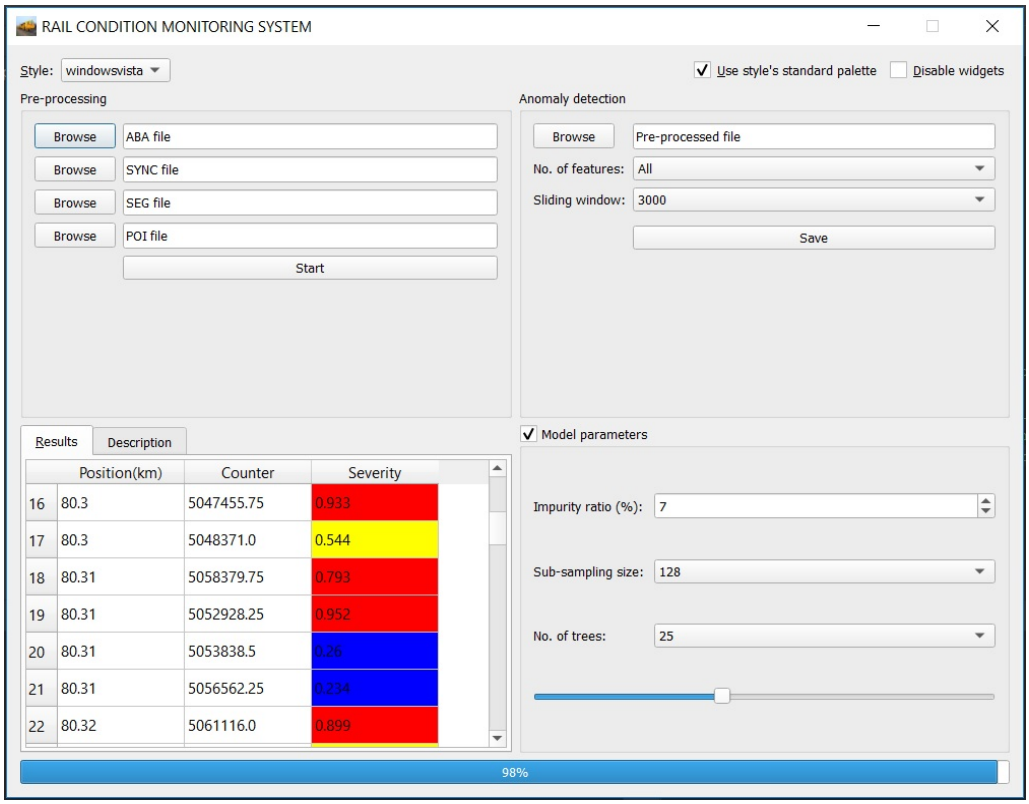


FIGURE 6.1: Design of Graphical User Interface (GUI) for rail CMS

6.2 Inputs and Outputs

The model requires several input files for pre-processing, anomaly detection and analysis with constraints on their format and file extension. For pre-processing the required files are raw ABA data file, Sync file, POI file, and Seg file. However, for anomaly detection, a pre-processed ABA data file is required. Moreover the model parameters that are required for anomaly detection, can be adjusted by the operator using the interface. A detailed description of the above mentioned input files are discussed in chapter 3 of the thesis.

6.2.1 Input files

The information regarding the formats and file extensions of the input files of the rail CMS are given as follows:

- **ABA data:** ABA is the core input to the rail CMS, used for anomaly detection. The ABA as an input must have H5 file extension. An H5 file is a data file saved in the hierarchical data format (HDF). It contains multidimensional arrays of scientific data. In order to perform the pre-processing, ABA data must be loaded into the system. In case, a pre-processed data file is already available, then the maintenance operator should load the pre-processed file for anomaly detection.
- **SYNC file:** The sync file should be given as input to the model in MS Excel file with CSV file extension. It is required to synchronize the ABA data with its external counters. Besides that, it provides information to filter out unsynced ABA data. Therefore the sync file is essential to perform the pre-processing.
- **POI file:** It should be inserted into the system as MS Excel format having CSV file extension. This file is required to obtain location information for ABA during the pre-processing.
- **SEG file:** The SEG file is feed into the model as MS Excel file with CSV format. This file is essential for identifying the direction and operational mode of inspection train in the ABA data during pre-processing.

6.2.2 Output files

There are two main outputs of the system (i) the pre-processing output and (ii) the anomaly detection output.

- **Pre-processing output:** The output of the pre-processing phase is saved in the user selected local directory in HDF format with h5 file extension. The pre-processed data is filtered, synchronized and transformed. The output from the pre-processing step is the input to the anomaly detection module.
- **Anomaly detection output:** The output of the anomaly detection model is saved in the local machine in a MS Excel file with CSV file extension. It contains the external counters, anomaly score and location (km) of the detected anomalies in ABA data.

6.3 User guidelines

The following are the instructions for the users to interact with the rail CMS and control some of the input parameters.

- The layout shown in figure 6.1 consists of four segments namely pre-processing, anomaly detection, model parameter settings, and results segment.
- By clicking the **Browse** button in the pre-processing segment, the user can load the required data files according to the given labels in front of each button.
- Once the files with correct formats and extensions are successfully loaded, then the pre-processing can be initiated by clicking the **Start** button.
- As soon as the pre-processing is completed, a **Save** button will appear which is used to store the pre-processed file in the local directory.
- In case the pre-processed file is already available, the user can straight away go for the anomaly detection process.

- In the anomaly detection segment, the users need to load the pre-processed file and set the values for sliding window and the intended ABA features. Besides that, the users are required to set the parameters of the model in the model parameters segment. The anomaly detection process can be initiated by clicking the **Start** button in the anomaly detection segment.
- The location, counters, and severity of the detected anomalies are displayed in the table in the results segment of the layout as soon as the anomaly detection process is completed. The generated results of anomaly detection model can be stored in a MS Excel file with CSV format in local storage by clicking the **Save** button in the anomaly detection segment.

Chapter 7

Discussion and Conclusions

This chapter presents the final discussion on the project, draws conclusions of the thesis report and provides direction for further enhancement in the project. The potential areas for improvements in the system are discussed in section 7.1. The conclusions of the project is presented in section 7.2. In section 7.3, the recommendations are given for further research and development in the data-driven rail maintenance project.

7.1 Discussion

The rail CMS is developed keeping in view the requirements at the stakeholder and the system level. Validation and testing of these requirements are done using the V-model approach, discussed in chapter 2. The purpose of the design project is to detect irregularities in train ABA data. The detection capability of the rail abnormality of the applied solution shows promising results. However the number of mishits and false alarms yielded by the model for all experiments with various features still asks for improvement in various segments in the ML pipeline. Each component in the pipeline can be replaced by its counterpart technique to achieve better hitrate and decrease the number of false alarms. This can be achieved in

an iterative manner on the basis of trial and error, because no ML approach exists that solves each problem in the best way. The choices made regarding the various techniques, algorithms and parameters for the entire process are based on previous experience and domain literature.

Calibration: The calibration of ABA data on channel *A* and channel *B* is one potential area that can be improved. This step is highly critical as it is the entry point to the anomaly detection model and data getting distorted at this stage would seriously affect the final outcome. In this work a transformed value is calculated using ABA data in *X* and *Z* axes. The tri-axial accelerometer data can be used in alternative ways to improve the end results, such as rotating the data from one sensor with an angle to coincide it with data from other sensor. For that, the angle of rotation must be known otherwise it has to be done in an iterative manner with a random selection of rotation angle. After each iteration the similary between the data of both sensors need to be checked. Using acceleration data in *X* axis only is also an option, although it will not represent the horizontal acceleration of the train axlebox because of misalignment.

Feature engineering: Signal processing is a critical part in the whole process and it carries high potential for improvement in the rail CMS. In this project, a set of statistical features is used by performing time-domain analysis of the ABA signal. Other types of features can be investigated by using frequency and time-frequency domain analysis specially in case of using a supervised ML approach for rail defect detection. The issue with selection of the optimal features and dimensionality reduction can be handled using methods like principal component analysis (PCA) and singular value decomposition (SVD). Moreover, the features can be selected based on their physical connection with the rail defects. Besides that, the size of the sliding window during feature extraction is also significant for improving the

performance metrics of the model.

Variable train speed: The current design of the rail CMS does not consider the factor of the train speed while processing and detecting anomalies in the data. The vibration level of the axle box of the train changes with variation in train speed. Since the train speed is varying througout the passage, not incorporating the factor of speed on ABA can have negative impact on the model outcome. Moreover, ignoring the speed factor creates problems while setting a fixed size sliding window during feature extraction. The sliding window size is kept constant assuming a constant speed througout the inspection campaign. It defines the length of the target area on rail track to be monitored for anomaly detection. When there is a variation in the speed, the targeted track length also changes accordingly because of the fixed window size. Ultimately the calculation of performance metrics of the model are negatively affected. A proper solution is needed to deal with the train speed variation. The flexibility of the designed pipeline allows to incorporate the impact of train speed. One way to do it, is to normalize ABA data with train speed, but the relation between speed and amplitude of ABA is not known. Therefore normalizing the data can distort the original information in the data. Hence, normalization is not the right and safe option. Another way to handle the impact of train speed is to make categories of ranges of train speeds such as below 40 km/h, from 40 km/h to 60 km/h and above 60 km/h. It has been reported in literature that the low train speeds do not have high significance in detection of rail defects using ABA data, while speeds around 70 km/h generate more valuable information regarding rail condition. Considering these research findings, the low speed ABA data can be filtered out, which is normally found near the rail stations and the switches. Only the data above 40 km/h can be processed and analyzed for defect detection. The high speed data are to be categorized and treated separately by the model. Each

detected anomaly will then be associated with its respective train speed.

Anomaly detection model: The parameters used for anomaly detection model such as sub-sampling size, contamination in data, and number of iTrees have an impact on the model output. The contamination ratio is significant because it gives a prior information to the model about the percentage of the outliers in the data. A higher contamination value would expand the threshold for data samples to be outliers and will result in larger number of anomalies and vice versa. Increasing the impurity ratio improves the hit-rate but it also increases the number of false alarms which is undesirable.

Validation: Validation of the model is done using rail images, however it would have been ideal if true outputs of ABA were available and used for the model validation. Besides that, the rail images require manual labelling of the defects which is laborious and can be erroneous as the early stage defects are not always clearly visible in the images. Doing so, the outcome of the model cannot be validated in a true sense, because training of the model is done on one type of data and the validation is performed on completely different data.

Channel and Passage comparison: Analysis on the channel comparison and the measurement train passages is carried out in order to assess the feasibility of ABA. The outcome for passage comparison is convincing because anomalies are found to be repeated in most cases at same location for multiple passages. It means that the abnormal behavior of ABA at certain points on rail is not random. The consistency shown in anomaly repetition reveals that ABA is potential candidate for representing rail irregularities. However, only the detection of the irregularities related to the rail defects is important. In channel comparison, a considerable number of anomalies were found on both sides of rail while a smaller amount of anomalies

were appeared just on one side of the rail. It can be said that severe rail abnormalities affect the ABA on both sides while incipient rail defects generate certain patterns in ABA on one side of rail.

Defects severity estimation: The severity analysis that is discussed in chapter 5, is done in pursuit of exploring the correlation between the defect crack size and ABA anomaly score. It is presumed that highly severe defects create high energy ABA signals and vice versa. The head-checks crack size and the corresponding anomaly score is plotted in figure 5.3 to show the correlation between these two variables. The correlation between crack severity and anomaly score is found to be rather random and inconsistent. Further analysis needs to be done using more data for various rail tracks that probably yield better correlation between crack severity and ABA abnormality. The correlation is important, as it can help in crack size estimation by feeding the anomaly score to the regression models. Based on the success of this analysis, each detected anomaly will be associated with the estimated crack size.

Rail defects classification: Labeling rail defects such as head-checks and squats can be done using the reports of EC and US testing reports . These reports contain information such as type, size, and location of defects on rail. Although the labelling task is hard and laborious, it will help in shifting the problem solution from an unsupervised to a supervised ML domain. Labelled data are highly vital for ML applications as then the models can be trained in a supervised manner for the defined classes in the data. Calculation of performance metrics and model optimization can be directly performed because of the availability of ground truth information. The models that are trained using a supervised approach, learn the patterns in the data for these defects. The trained models can identify the head-checks and squats in newly acquired ABA data. Based on the prediction accuracy, the model's

parameters can be tuned accordingly to obtain the best outcome. Moreover, the rail images based validation will be no more required in the supervised way of training models. The validation will rather be performed during the training phase of the model using the actual data labels. Supervised training of the models can be performed using either classical ML techniques or deep learning (DL) techniques depending on the input type of the data. One positive characteristic of the DL is that these techniques do not require features to be extracted beforehand. The DL models extract and optimize the features themselves and identify various classes in the data. The big question regarding the ABA labelling using EC and US testing reports is the reliability of these reports.

7.2 Conclusion

The main goal of the project was to come up with a solution to the design problem, presented in chapter 1. The objective was to develop a condition monitoring system for rail infrastructure in pursuit to detect incipient rail defects that ultimately improve time and cost of rail maintenance. The conclusions from this work are drawn at the same levels where the requirements were defined. At the system's level the following concluding points are drawn:

- The GUI of the rail CMS is supportive and friendly for user interaction. The usage guidelines are provided in this report. This fulfills the requirements SH1, and SYS8.
- The developed rail CMS can be used for anomaly detection on various rail tracks, which fulfills the requirement SH2.
- The rail CMS takes the ABA as the main input for anomaly detection according to the requirement SYS9.

- The rail CMS is developed using python to enable its integration with python based systems. It is done in fulfilment of the requirement SYS6.
- The rail CMS is flexible to modification and bug fixing, which fulfills the requirement SYS13.
- The rail CMS enable the users to save the output of anomaly detection in local storage as per requirement SYS10.

Considering the requirements at the stakeholder's level, the following conclusions are drawn after completing the design process:

- The rail CMS generates clear human interpretable information in the form of location and severity of the detected anomalies on rail according to the requirement SH4.
- In order to fulfill the requirement SH3, further improvements are required in the rail CMS as given in section 7.3 to help in decision making regarding initiation of rail maintenance.

7.3 Recommendations

The following are the recommendations to move forward in the data-driven rail maintenance project:

- Including the train speed factor in the designed ML pipeline.
- Rail defects detection and classification using a supervised ML approach by providing labelled ABA data.
- Severity estimation of the rail defects.

Bibliography

Abe, Naoki, Bianca Zadrozny, and John Langford (2006). "Outlier detection by active learning". In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 504–509.

Assis Boldt, Francisco de et al. (2015). "Fast feature selection using hybrid ranking and wrapper approach for automatic fault diagnosis of motorpumps based on vibration signals". In: *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*. IEEE, pp. 127–132.

Bonnema, G Maarten (2014). "Communication in multidisciplinary systems architecting". In: *Procedia CIRP* 21, pp. 27–33.

Breunig, Markus M et al. (2000). "LOF: identifying density-based local outliers". In: *ACM sigmod record*. Vol. 29. 2. ACM, pp. 93–104.

Caesarendra, Wahyu and Tegoeh Tjahjowidodo (2017). "A review of feature extraction methods in vibration-based condition monitoring and its application for degradation trend estimation of low-speed slew bearing". In: *Machines* 5.4, p. 21.

Esveld, Coenraad (2001). "Modern Railway Track, 2nd Editon". In: *Delft university of Technology*.

He, Zengyou, Xiaofei Xu, and Shengchun Deng (2003). "Discovering cluster-based local outliers". In: *Pattern Recognition Letters* 24.9-10, pp. 1641–1650.

Heng, RBW and Mohd Jailani Mohd Nor (1998). “Statistical analysis of sound and vibration signals for monitoring rolling element bearing condition”. In: *Applied Acoustics* 53.1-3, pp. 211–226.

Islam, Md Rashedul, Sheraz A Khan, and Jong-Myon Kim (2015). “Maximum class separability-based discriminant feature selection using a GA for reliable fault diagnosis of induction motors”. In: *International Conference on Intelligent Computing*. Springer, pp. 526–537.

ISO/IEC/IEEE29148 (2011). *Systems and software engineering – Life cycle processes – Requirements engineering*. Standard. Geneva, CH: International Organization for Standardization.

Liu, Fei Tony, Kai Ming Ting, and Zhi-Hua Zhou (2008). “Isolation forest”. In: *2008 Eighth IEEE International Conference on Data Mining*. IEEE, pp. 413–422.

Magel, Eric et al. (2008). “Traction, forces, wheel climb and damage in high-speed railway operations”. In: *Wear* 265.9-10, pp. 1446–1451.

Marino, Francescomaria et al. (2007). “A real-time visual inspection system for railway maintenance: automatic hexagonal-headed bolts detection”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 37.3, pp. 418–428.

NI (2019). *http://www.ni.com/product-documentation/3727/en/*. Standard. The NI TDMS File Format, National Instruments.

Rousseeuw, Peter J and Katrien Van Driessen (1999). “A fast algorithm for the minimum covariance determinant estimator”. In: *Technometrics* 41.3, pp. 212–223.

Thomas, Hans-Martin, Thomas Heckel, and G Hanspach (2007). “Advantage of a combined ultrasonic and eddy current examination for railway inspection trains”. In: *Insight-Non-Destructive Testing and Condition Monitoring* 49.6, pp. 341–344.

Veit, Peter (2007). “Track Quality-Luxury or Necessity?” In: *Railway technical review/RTR special* July, pp. 8–12.