Master's thesis

Ole Sørheim

Artificial Intelligence-based Algorithm for Incipient Fault Diagnosis of Salient Pole Synchronous Generator using Multiple Sensor Fusion

Master's thesis in Energy and Environmental Engineering
Supervisor: Arne Nysveen
Co-supervisor: Hossein Ehya
June 2021

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electric Power Engineering

Ole Sørheim

Artificial Intelligence-based Algorithm for Incipient Fault Diagnosis of Salient Pole Synchronous Generator using Multiple Sensor Fusion

Master's thesis in Energy and Environmental Engineering
Supervisor: Arne Nysveen
Co-supervisor: Hossein Ehya
June 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electric Power Engineering

Abstract

This thesis is a continuation of the previous work within fault detection of synchronous generators at NTNU. Hydropower represents roughly 90% of all energy produced in Norway. With an increasing demand for clean energy, the sustainability of hydropower generators is a prime concern. The detection of incipient faults in hydropower generators is vital in ensuring reliable and sustainable energy production. This thesis investigates the use of machine learning and multi sensor-fusion in on-line condition monitoring of synchronous generators for the propose of early discovery of incipient faults. The work is focused on the detection of Inter-Turn Short-Circuit (ITSC) and Static Eccentricity (SE). Moreover, the application of signal processing tools to extract useful patterns as an input to the artificial intelligence (AI) algorithm has been studied. The thesis shows that by applying sensor-fusion of vibration and stray magnetic field to a suitable machine learning methodology, an accuracy of 95% in detection of ITSC can be achieved.

The dataset used in the machine learning algorithms was extracted from a 100kVA salient-pole synchrnous generator at the NTNU facilities. This generator was designed to represent a typical hydropower generator and could be imposed with various faults for case study. As a part of the work with machine learning, measurements of vibration, stray magnetic field, stator voltage, and stator current was extracted for various operating conditions. The generator was imposed with ITSC and SE of various degree during measurements. Features were extracted from the measured signals using Fast Fourier Transform (FFT), Discrete Wavelet Transform (DWT) and the TSFRESH-algorithm. The features were filtered by the Random Forest classifier and TSFRESH-algorithm yielding approximately 170 and 750 relevant features, respectively. The features extracted through FFT and TSFRESH showed the highest importance in detecting faults, while DWT-features showed a subpar performance.

A multiple of machine learning algorithms have been evaluated and compared, including Logistic Regression, K-Nearest Neighbour, Support Vector Machine, XGBoost, Artificial Neural Network and ensemble classifiers. The aforementioned classifiers were optimized using a grid-search of the most relevant hyperparameters. Different splitting strategies for training and test data have been analysed, and a proposed methodology has been presented. Sensor-fusion of stray magnetic field and vibration was performed based on the waterfall fusion model. The datasets containing stator voltage and current was not sufficient to be used in machine learning and the result using this dataset was questionable. However, the thesis shows that by utilizing only stray flux and vibration in sensor-fusion, the prediction performance of the algorithms improved.

The dataset extracted from the laboratory generator was severely imbalanced, with the minority class representing a healthy generator. The training set was balanced using SMOTE-ENN, an algorithm that creates synthetic samples of the minority class based on the relative distance between samples. Due to the imbalance the most relevant metrics for evaluation of the algorithm were ROC AUC and specificity. ROC AUC describes the classifiers ability to distinguish classes. The proposed algorithm had an specificity of 0.9440 and a ROC AUC of 0.9475 in detection of ITSC, meaning out of all the negative classes 94.4% was correctly classified. For detection of static eccentricity all classifiers had a perfect score. It was concluded that the perfect score was not a result of data leakage. An evaluation of the generalizability of the algorithm has been performed by analysing the performance on datasets obtained from two industrial generators.

Sammendrag

Vannkraft representerer omtrent 90% av all energi som produseres i Norge. Med et økende behov for ren energi er bærekraften til vannkraftprodusenter et hovedanliggende. Det er viktig å oppdage begynnende feil i vannkraftgeneratorer for å sikre pålitelig og bærekraftig energiproduksjon. Denne avhandlingen er en videreføring av det tidligere arbeidet innen feildeteksjon av synkrongeneratorer ved NTNU og undersøker bruken av maskinlæring og multisensorfusjon i tilstandsovervåking av synkrongeneratorer for å detektere feil i generatorer. Arbeidet er fokusert på deteksjonen av kortslutningsfeil i rotor viklinger og statisk eksentrisitet. Videre har anvendelsen av signalbehandlingsverktøy for å trekke ut nyttige mønstre for bruk i kunstig intelligens blitt studert. Oppgaven viser at ved å anvende sensorfusjon av vibrasjon og spredefelt sammen med en egnet maskinlæringsmetodikk, kan man oppnå en treffsikkerhet på 95% i deteksjon av kortslutningsfeil.

Datasettet som ble brukt i maskinlæringsalgoritmene ble hentet fra en laboratoriegenerator ved NTNU-anleggene. Denne generatoren var designet for å representere en typisk vannkraftgenerator og kunne påføres forskjellige feil for casestudie. Som en del av arbeidet med maskinlæring ble målinger av vibrasjoner, spredefelt, statorspenning og statorstrøm hentet ut ved forskjellige driftsforhold. Generatoren ble påført kortslutnings feil og eksentrisitet av ulik grad under målingene. Kjennetegner på feilene ble hentet fra de målte signalene ved hjelp av Fast Fourier Transform (FFT), Discrete Wavelet Transform (DWT) og TSFRESH-algorithmen. Disse kjennetegnene ble filtrert basert på relevans for deteksjon av feil ved hjelp av Random Forest-klassifisereren og TSFRESH-algoritmen, som henholdsvis ga 170 og 750 relevante kjennetegn. Kjennetegnene hentet gjennom FFT og TSFRESH viste høyeste betydning for å oppdage feil, mens DWT-Kjennetegnene var mindre viktig.

Et mangfold av maskinlæringsalgoritmer har blitt evaluert og sammenlignet, inkludert Logistisk regresjon, K-nærmeste nabo, Support Vector Machine, XGBoost, Artificial Neural Network og ensemble-klassifiserere. De nevnte algorithmene ble optimalisert ved hjelp av et rutenett-søk av de mest relevante hyperparametrene. Forskjellige splittingsstrategier for trening og testdata har blitt analysert, og en foreslått metodikk har blitt presentert. Sensorfusjon av spredefelt og vibrasjon ble utført basert på fossefalls-fusjonsmodellen. Datasettene som inneholdt statorspenning og strøm var ikke tilstrekkelig til å bli brukt i maskinlæring, og resultatet ved bruk av disse datasettetene var tvilsomt. Avhandlingen viser imidlertid at ved å bruke spredefelt og vibrasjon i sensorfusjon, ble prediksjonens ytelse til algoritmene forbedret.

Datasettet hentet fra laboratoriegeneratoren var sterkt ubalansert, med en minoritetsklasse som representerte en frisk generator. Som sådan ble balanseprosedyrer implementert og treningssettet ble balansert ved hjelp av SMOTE-ENN. På grunn av ubalansen var ROC AUC og spesifisitet de viktigst for evalueringen av algoritmene. Den foreslåtte algoritmen hadde en spesifisitet på 0,9440 og en ROC AUC på 0,9475 i deteksjon av kortslutningsfeil. For deteksjon av statisk eksentrisitet hadde alle algorithmene en perfekt poengsum. Det ble konkludert med at den perfekte poengsummen ikke var et resultat av datalekkasje. En evaluering av generaliserbarheten av algoritmen har også blitt utført ved å analysere ytelsen på datasett hentet fra to industrielle generatorer.

Preface

This Master's thesis represents the culmination of my 5-years of studying engineering at the Norwegian University of Science and Technology (NTNU). The work was conducted in the spring of 2021 at the Department of Electric Power Engineering under supervision of Arne Nysveen and Hossein Ehya.

The thesis investigates the usage of machine learning and multi-sensor fusion in the field of fault detection of synchronous generators. The basis for the work is a state of the art laboratory generator at NTNU. I am grateful for the opportunity to work with machine learning and artificial intelligence, and the work has been both rewarding and challenging. The exceptional facilities at NTNU has formed the foundation for a high quality report and I hope the thesis does both NTNU and my supervisors justice.

I would like to express my sincere gratitude towards my supervisors Arne Nysveen and Hossein Ehya for providing unparalleled support throughout this thesis. Hossein Ehya's expertise and profound knowledge in synchronous generators and condition monitoring has been indispensable for this thesis. I would like to thank you for all the time you have put into guiding me through this work, it has been highly appreciated. I'd also like to thank Arne Nysveen for the opportunity to work with such a rewarding thesis and for the help in finalizing the work. I wish you both all the best in your future work. Finally, I'd like to thank my family and friends for providing support and illuminating discussions throughout my years at NTNU.

Contents

List of Figures

List of Tables

Acronyms

AI Artificial Intelligence. 1, 9, 10, 62

ANN Artificial Neural Network. I, 17

BDB Broken Damper Bar. 5

CV Cross-Validation. 10

CWT Continuous Wavelet Transform. 23, 24

DFT Discrete Fourier Transform. 22

DWT Discrete Wavelet Transform. I, 23–25, 34, 37, 38

EDA Exploratory Data Analysis. 42

EMD Empirical Mode Decomposition. 24

FFT Fast Fourier Transform. I, 3, 22, 23, 34, 38

FPR False Positive Rate. 55

HHT Hilbert-Huang Transform. 3

HVCB High-Voltage Circuit Breaker. 6

HWE Hierarchical Wavelet Energy. 25

ITSC Inter-Turn Short-Circuit. I, 3, 18, 24, 29, 30, 37, 38, 42, 50, 61, 67, 68, 71

IWE Instantaneous Wavelet Energy. 24

KNN K-Nearest Neighbour. I, 14, 15, 42, 64

MLP Multilayer Perceptron. 17

OOB Out-Of-Bag. 16

OSS Original Sample Series. 34, 63

RF Random Forrest. 5, 18, 48, 49

ROC Receiver Operating Characteristics. 21

ROC AUC Area Under Curve the ROC Curve. 21, 48

RSS Reduced Sample Series. 34, 37, 63

RWE Relative Wavelet Energy. 24, 25

SE Static Eccentricity. I, 3, 29, 30, 35, 38, 41, 42, 50, 61, 67, 68, 71

SMOTE Synthetic Minority Oversampling Technique. 19

SPSG Salient-Pole Synchronous Generator. 18, 69

STFT Short Time Fourier Transform. 6

SVM Support Vector Machine. I, 5, 6, 12, 51

TPR True Positive Rate. 55

TSFRESH Time Series Feature Extraction Based on Scalable Hypothesis Tests. VII, 25, 26, 37, 50

TWE Teager Wavelet Energy. 24, 25

XGBoost eXtreme Gradient Boosting. 18

Chapter 1

Introduction

1.1 Background

Synchronous generators represents a large portion of the power generating units in the modern world. In Norway approximately 90% of the energy produced comes from hydropower, which mainly consists of large synchronous generators [1]. The dominance of synchronous generators in the Norwegian power system places a high demand on the sustainability of these machines in order to maintain reliability in the electrical grid. An unplanned outage of power plants is one of the most critical concerns in the electrical power industries.

Synchronous generators are one of the most complex and expensive equipment in a power plant. An unscheduled outage of the generators due to a fault causes a large financial loss. In addition, maintenance and repair costs are high due to the complexity of the machines. The discovery of an aggravating fault in the incipient stages can significantly reduce the repair costs and prevent an unscheduled outage. Periodic maintenance of electrical machines has been shown to reduce the maintenance cost by up to 60%, indicating a large economic gain in locating faults at the early stages [2].

Most faults in synchronous generators cause some form of asymmetric flux density within the machine. This asymmetry has repercussions throughout the dynamics of the machine. By monitoring certain signals from the machine, these repercussions can be located and used to identify faults in the machine. On-line condition monitoring is the method of monitoring fault indices in the signals in real-time for the purpose of incipient discovery of aggravating faults.

In recent years, research has centered around the use of machine learning and other artificial intelligence (AI) techniques in fault detection. With a wide arrange of classifiers, researchers has achieved well above 90% accuracy in detecting faults in electrical machines [3, 4, 5, 6, 7]. Multi-sensor fusion is a new field of research in condition monitoring of electrical machines and utilizes the signals from multiple sensors to improve the predictive performance of machine learning algorithms. Induction machines are vastly overrepresented in the number of electrical machines in the market and, naturally, more research has been conducted on such machines. As such, there is a distinct gap in the research of multi-sensor fusion based algorithms for large synchronous generators. The exceptional laboratory setup at NTNU forms a solid foundation for research into faults in synchronous generators, research that could prove highly beneficial for the hydropower industry.

This Master thesis is a continuation of the recent master's thesis conducted at the department of electrical power engineering at NTNU [8]. In [8] the use of machine learning in fault detection based on air gap magnetic field was investigated. The performance of the algorithm presented in the thesis was underwhelming due to both lack of sufficient training data as well as other conditions. The purpose of this thesis is to improve the algorithm presented in [8] by utilizing multi-sensor fusion. In addition, new measurements are to be taken to increase and improve the training data. Sensor fusion are to be conducted based on stray magnetic field, vibration, and stator current and voltage. Air gap magnetic field has not been considered in this thesis as it is an invasive method and stray magnetic field has been proven to provide an equal or better performance in fault detection systems [9].

1.2 Objectives and scope of work

This thesis is a continuation of the specialization project conducted in the autumn of 2020. According to guidelines on self-plagiarism at NTNU, it is hereby stated explicitly that some of the content is adopted from the specialization project. The following sections are either verbatim or adapted from the specialization project preceding this thesis; Section 1.1, adapted from; Section 2.2, verbatim; Section 2.7.1, adapted from; Section 3.1, verbatim ; Section 3.2, adapted from [9].

The objectives of the thesis are as follows:

- Obtain measurements of stray magnetic flux, vibration, current and voltage from the NTNU laboratory generator.
- Calculate features from the obtained signals which correlate to the state of the machine.
- Develop a machine learning algorithm based on the algorithm presented in [8] that uses multi-sensor fusion for detection of inter-turn short-circuit and static eccentricity.
- Evaluate performance and perform simple optimization of said algorithm.
- Investigate the possibility of a generalizable condition monitoring system for use on synchronous generators in the industry which shares the characteristics of the laboratory generator.

1.3 Outline

Chapter 1 - Introduction: Introduces the objectives and scope of the Thesis

Chapter 2 - Theoretical Background: Outlines the theoretical background of the thesis, with focus on machine learning and the classifiers used in the thesis.

Chapter 3 - Laboratory Setup: Describes the laboratory setup used to extract measurements.

Chapter 4 - Methods and Results: Outlines the methodology and results used in designing a machine learning algorithm for fault detection. Includes a brief discussion of results as they are presented.

Chapter 5 - Discussion: Presents the discussion of the methodology and results in light of the theory presented in chapter 2.

Chapter 6 - Conclusion: Presents a summary of the most important conclusions of the thesis.

1.4 Previous Work

Following the development of the laboratory generator at NTNU and the cooperation with HydroCen, several master's thesis have been conducted in the field of fault detection of synchronous generators within the department of electrical engineering at NTNU. The design of the laboratory generator at NTNU created an exceptional opportunity to study the effects of faults in electrical machines. The signatory thesis draws on the work conducted by two fellow students in their final year at NTNU and is a direct continuation of [8, 9, 10]. Following is a brief summary of the previous work.

In the specialization project prior to this thesis an investigation of stray flux in fault detection was conducted [9]. Only the impact of ITSC was analysed. The report concluded that stray flux is well suited for use in fault detection, both for detecting faults as well as to detect severity. The analysis was performed with both FFT and HHT, with the former providing several advantages. Loading was found to have a minor impact on the proposed indices. In addition, fault detection using stray flux was found to be comparatively better than using the air gap magnetic flux, mostly due to the non-invasive nature of the method. Stray flux was found to contain equal or greater information about the state of the machine relative to air gap flux. The analysis was performed on simulation data and verified on experimental results.

In [10] a vibration analysis based on simulation data was conducted. The purpose of the thesis was to identify fault signatures in the vibration signal of the lab generator. Both Inter-Turn Short-Circuit (ITSC) and Static Eccentricity (SE) was investigated. It was concluded that both ITSC and SE can be detected based on an FFT analysis of the vibration. The thesis found a near linear relationship between certain harmonic amplitudes and fault severity, meaning fault severity can be deducted by comparing the faulty signal with the healthy case. In addition, it is claimed that fault type can be distinguished by evaluating the vibration severity, as ITSC causes significantly higher vibration in the machine. The harmonics proposed for fault detection can be found in [10]. Most fault identifying harmonics were subharmonics of the fundamental frequency of the machine, with the only higher order harmonics being 100Hz and 200Hz. The loading of the machine was found to have insignificant impact on the frequency spectrum. The proposed method of fault detection was based entirely on simulation data and as such, a verification on data obtained from a real machine is needed.

In the master's thesis of T. N. Skreien a machine learning algorithm for fault detection based on air gap magnetic field measurements was developed [8]. For purpose of further work, the scope of the thesis was broad and the performance of several classifiers was compared. The analysis was performed on measured air gap flux, however, due to the Covid-19 situation new measurements could not be conducted. As such, the thesis was limited to data sets gathered previously in work related to different projects at NTNU. Due to this data constraint, and other factors, the proposed algorithm had an underwhelming accuracy of about 85%. The signatory thesis is formed with a basis in the code developed in [8] with the purpose of improving the accuracy of the machine learning algorithm by the addition of multi-sensor fusion. As such, most coding done in this thesis is an adaption of the code used in [8].

Chapter 2

Theorethical Background

2.1 Multi-Sensor Fusion

As found in the literature review conducted in [9], fault detection of electrical machines are a major field of research and multiple different techniques have been developed over the years. Based on the signal processing tool and the type of signal extracted the fault detection techniques are faced with different challenges and strengths. The idea of multi-sensor fusion arise from combining multiple techniques to cover the weaknesses of the different methods and thus improve the robustness and accuracy in fault detection. This synergistic effect arise from combining the collective information from multiple sensors into a mutual representation of the system. Multi-sensor fusion have been successfully employed in the fields of robotics, defense, equipment monitoring, biomedical engineering and transportation systems [11]. Following is a brief literature review on multi-sensor fusion in fault detection of electrical machines.

In [12], an overview of multi-sensor fusion techniques are provided. The methods of sensor fusion presented in the paper can be summarized in three categories: *complementary* fusion, *signal based* fusion and *decision level* fusion. In complementary fusion each signal provide some information about the system which are used complementary to each other. A common implementation of complementary fusion is to use signal processing tools to extract features from the different sensors, which are then stored in a single feature vector. In signal based fusion, the different signals are combined into a single signal that provide a better representation of the system. There are various ways of combining the signal, some of which are based on weighted averaging, kalman filter, neural networks or non-linearly averaging [12, 13]. Kalman filters are frequently used for sensor fusion in robotics and drone technology. Decision level fusion combines several sub-decisions or features to yield a final decision [12]. A popular method for decision level fusion is through Dempster-Shafer Evidence Theory (D-S theory). D-S theory sensor fusion finds widespread use in the field of human-robot interaction [12].

Support Vector Machine (SVM) and Random Forrest (RF) has been successfully used to detect BDB and eccentricity faults in an induction machine by means of multi-sensor fusion [5]. The researchers used frequency- and time-domain features from vibration and current measurements to train a SVM classifier. Feature selection through random forest was used to select the most relevant features for the fault classification problem. The paper shows that by utilizing multi-sensor fusion the performance of the algorithm significantly increased and also shows promising results in discrimination of severity and simultaneous faults by means of one-vs-all classifiers and multi-sensor fusion.

In [4] the researches presents a review of the trends and challenges in intelligent condition monitoring of electrical machines using machine learning and makes a point that ML techniques are not a novelty. Intelligent condition monitoring methods are mostly used in combination with traditional fault detection methods to increase robustness and performance of the system. The main challenges pointed out in the paper revolves around the data set. For AI techniques, the train/test data set is of high importance, and must be of a certain quality and size to obtain valid results. This contribute to a major challenge in the industry where adequate datasets of the machine states are hard or impossible to extract.

In [13], vibration, sound, current, voltage, and temperature are used in to detect motor faults through Support Vector Machine (SVM). The researchers fused the output of multiple sensors into a single signal by non-linearly averaging the multiple measurements (signal based fusion). The signals was acquired from a brushless DC motor. This fused signal was then analysed by Short Time Fourier Transform (STFT) to extract features. The proposed methodology of sensor fusion SVM based on STFT pre-processing reached an accuracy of 95% on the test set.

Sensor fusion and semi-supervised deep learning have been used effectively to detect gear faults in induction machines [14]. The researchers used measurements of stator current, vibration, torque and sound. Sensor fusion was performed by segmenting the measurements and calculating a wide range of features on each of the segmented samples. The signal processing tools included FFT, WT and EMD. In order to adapt to a high-dimensional feature space with few labeled samples, the researches used a semi-supervised deep ladder network (SSDLN) for fault detection. The semi-supervised deep learning algorithm outperformed other supervised learning algorithms, convolutional neural network and low density separation, with an overall accuracy of about 99.79%.

In [15], mechanical faults in HVCB are detected using multi-sensor fusion based on multiple vibration signals. The researchers used evidence theory for dealing with multi-sensor input, which allow for multiple evidence from different sources to be combined [12]. The proposed fusion method is built on Dempster-Shafer Evidence Theory (D-S theory) by incorporating entropy measures to determine weights for the different measurements. The proposed method of sensor fusion is shown to outperform traditional D-S theory sensor fusion. Multinomial logistic regression was used for fault classification. The paper includes a comparison of different fusion techniques as well as different classifiers, which shows the superiority of the proposed method. The proposed method boast a predictive accuracy of about 95%.

2.2 Faults in Electrical Machines

This entire section is adapted from the specialization project written by the signatory in the autumn of 2020 [9].

In this section the causes and consequences of faults in electrical machines are described. There are multiple possible faults in an electrical machine and the full description of all faults are beyond the scope of this report. Thus the faults mentioned in this section are limited to those most prevalent in synchronous generators, namely short-circuit (SC) of rotor or stator windings and eccentricity.

2.2.1 Eccentricity

Eccentricity means there is a non-uniform air gap length in the machine due to displacement of either the rotor with respect to the stator (static eccentricity), the rotational axis with respect to the rotor centre of mass (dynamic eccentricity) or a combination of both (mixed eccentricity), as illustrated in fig. 2.1. In static eccentricity, the air gap length is stationary and dependant on position, while in dynamic eccentricity the air gap length varies with time. In mixed eccentricity, there is both a stationary and a transient component in the air gap length. The non-uniform air gap length creates asymmetries in the air gap magnetic field due to changes in reluctance and magnetizing inductance [16]. A greater air gap length leads to a higher reluctance which in turn decreases the flux in the respective flux path. This leads to a decrease in the induced voltage and the magnetomotive force (mmf). On the other side, lower air gap length increases the flux and thus also increase the induced voltage and mmf. As such, the non-uniform air gap length creates an unbalanced magnetic pull (UMP) in the machine as a function of the differences in mmf. This UMP creates noise and vibration in the machine, which causes additional mechanical stress and can aggravate the fault.

Static eccentricity (SE) forms a stationary UMP due to the time-independent air gap length. It is reported that the mechanical stress of SE often leads to dynamic eccentricity (DE) due to failures in mechanical parts such as the bearing [17, 18]. Additionally, the UMP leads to added tension on the windings which can lead to insulation breakdown and short-circuit faults. It is therefore of great importance to discover eccentricity faults at the early stages due to the potential of escalating faults.

The induced electromotive force (emf) in the stator windings are directly related to the magnetic flux through Faraday's law, and harmonics in the magnetic flux induce harmonics in the voltage and current of the machine. Harmonics arise from the asymmetric flux due to faults such as eccentricity. These harmonics increase the copper and iron losses in the machine, which increases the temperature. Harmonics are also known to influence the torque ripple in electrical machines. Thus, the harmonics generated from in the flux decreases the performance of the machine and may increase vibration and temperature to such degrees that other faults are eminent. The main causes of eccentricity are summarised in table 2.1, [19, 18].

2.2.2 Short-circuit of rotor or stator windings

The windings of a synchronous generator are typically made up of multiple thin layers of insulated copper conductors. These conductors are insulated both from each other and the respective machine body. A short-circuit occurs as an effect of insulation breakdown

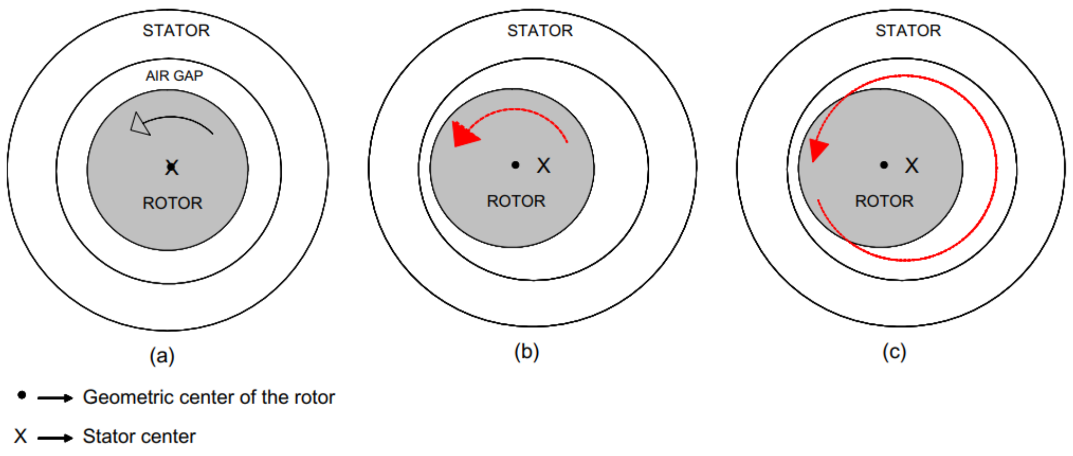


Figure 2.1: Types of eccentricity: (a) Healthy, (b) static eccentricity, (c) dynamic eccentricity [20].

Table 2.1: Main causes of eccentricity.

Type	Main cause
Static eccentricity:	Stator ovality Design/Assembly problems Misalignment
Dynamic eccentricity:	Bearing failure Mechanical resonance at the critical load Shaft bending

and can occur both between phase/ground and between phases. A short-circuit between individual conductors within a winding is called inter-turn short-circuit (ITSC) and are the most common SC-fault in electrical machines [21]. short-circuits most often occur in the field windings of the rotor in synchronous machines due to the higher mechanical stress [22].

The main cause of short-circuit is insulation breakdown. Insulation breakdown are a gradual process usually caused by excessive stress on the windings. This stress comes in terms of vibration, high temperatures and high voltage in the coils. Furthermore, the windings are subject to large forces from the magnetic fields and rotation of the machine. These magnetic and centrifugal forces cause wear and tear on the windings which can eventually lead to insulation deterioration [21].

When an ITSC in the rotor occurs, the current direction in the shorted turns are reversed, which creates an opposing magnetic field and mmf [22]. As such, ITSC effectively reduces the number of ampere-turns and decreases the magnetic flux and mmf of the respective pole. This results in an asymmetric flux density and unbalanced force distribution in the machine, i.e. UMP. As previously mentioned, UMP are an unwanted feature as it can aggravate faults and lead to additional faults in multiple parts of the machine. The decrease in flux leads to a reduction in the emf of the armature windings, which causes harmonics in the induced voltage. These harmonics are a factor of the fundamental mechanical frequency of the machine expressed in eq. (2.1),[23].

$$f_{fault} = \frac{p+k}{p} f_e \tag{2.1}$$

2.3 Machine Learning

Machine learning is a term used for computer algorithms that learn by experience or sampled data, and is a branch of artificial intelligence. A machine learning algorithm builds a model based on sampled data, referred to as *training data*, which is used to make predictions or decisions based on future data. A commonly known example of machine learning is email-filters. These algorithms builds a model to classify certain emails as spam based on prior emails, i.e. learns through processing a large database of emails containing both spam and non-spam.

Machine learning algorithms falls into mainly two categories, supervised- and unsupervised learning. Reinforcement- and semi-supervised learning are other categories of machine learning that will not be covered in this thesis. With supervised learning, training data contains target labels that specifies what class the sample belongs to. The training of these classifiers consists of fitting a model to the training data that minimizes some cost function based on the target labels. The cost function determines the weighting of different parameters, which can shift the model in terms of bias/variance (discussed later). Due to the dataset obtained in this thesis containing target labels for all samples, only supervised learning have been pursued.

Before diving into the different machine learning algorithms a fundamental trade-off needs to be addressed, the bias-variance trade-off. With any supervised learning algorithm, some parameters are chosen which determines how much emphasis should be put on individual observations. This is easily illustrated with regression as seen in fig. 2.2, where the datapoints represents the training data of a regression classifier. In fig. 2.2a the classifier are of a too low order, i.e. low emphasis on individual samples, to fit the data and the classifier would perform poorly (underfit). Because of the simplified model used to fit the data, the model has a high bias meaning a different training set are not likely to change the regression model significantly. In fig. 2.2b the regression model perfectly fits the training data, however, it would generalize poorly to new introduced data (overfit). An overfitted classifier is said to have high variance, and would typically perform exceptionally well on the training data but have low performance on the test set. To achieve a high performing classifier, both variance and the squared-bias should be minimized [24]. However, these quantities are often dependable, meaning a classifier that fits the training data extremely well will often generalize poorly, and vice versa. The trade-off involves giving the classifier enough slack, allowing it to tolerate some falsely classified observations, to have sufficient generalizability while maintaining high prediction accuracy.

2.3.1 Training, Testing and Cross-Validation

Training of AI refers to the processes of fitting a model to a data set, called the training set. In supervised learning the model is adjusted based on the performance on the training data, with the finalised model being the one that best matches the data to the target labels. Hyperparameters, parameters which predetermines the architecture of the algorithm, are not affected by training.

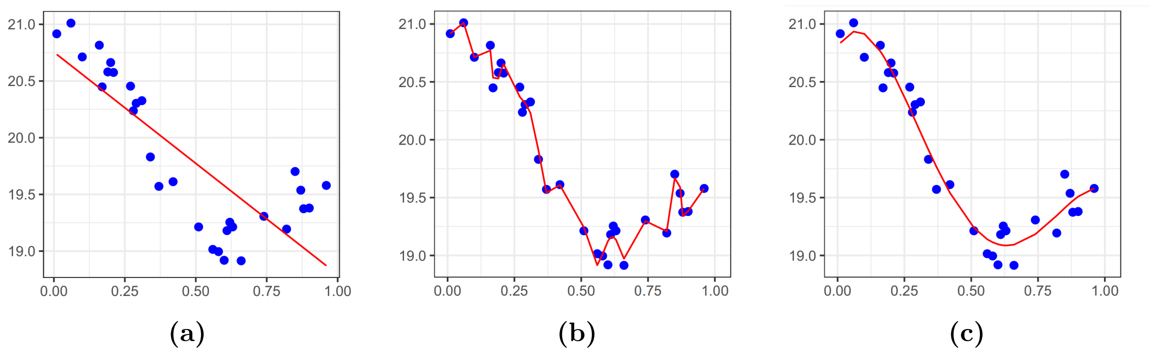


Figure 2.2: Overfitting/underfitting illustrated through regression: (a) underfitting, (b) overfitting, (c) optimal fit [25].

During training, the target labels in the training data are used to optimize the model. As such, the finalized model cannot be tested on the training data, otherwise data leakage would occur. Data leakage occurs when the training and testing of an algorithm occurs on the same data. An algorithm will always perform better on the data set on which the model was created, than that of new unseen data. Data leakage, also referred to as target leakage, results in an overly optimistic estimate of the algorithms performance. The data needs therefore to be separated into at least two independent sets, the training and testing set. The model can then be adjusted based on the training data and the performance can be evaluated based on the test data.

When dealing with only one estimator with fixed hyperparameters, two data sets would suffice. However, hyperparameters and the AI algorithm must be chosen based on the data set and the problem at hand. In such a case, two data sets would not suffice, as one cannot evaluate the performance of different algorithms and architectures on the training data, and if evaluated on the test set, data leakage would occur. This is usually dealt with by splitting the data into three or more sets: a training, a testing, and a cross-validation (CV) set. The CV-set serves as a separate test set to analyse the performance of different algorithms and hyperparameters while building the model. The model architecture is then chosen based on the performance on the cross-validation set, and the final algorithms performance is evaluated on the test set. This ensures that the reported performance of the AI is a realistic estimate of the performance on new data.

When splitting the data into several sets, the split itself impacts the performance of the algorithm. Data sets are typically non-uniform, thus certain splits might yield very good results, while others poor results based on which samples are put in each set. This phenomenon is called sampling bias. An often utilized method of decreasing sampling bias is k-fold cross-validation. k-fold CV can be used if the data set is to small be split into three sufficiently large sets or if the dataset is subject to a large sampling bias. k-fold cross-validation are done by splitting the training data into k subsets, then training the algorithm based on $k - 1$ subsets. The final subset is used to evaluate the performance of the algorithm across that individual fold. This is repeated k times such that each subset is used for validation once. The algorithm is then assessed based on the average performance across all folds [25]. The average performance across the k-folds give a more accurate depiction of the both the predictive performance and generalizability of the algorithm [8, 25]. The three splitting methods discussed are illustrated in fig. 2.3, fig. 2.4 and fig. 2.5.

Training
Testing

Figure 2.3: Training and testing split of data.

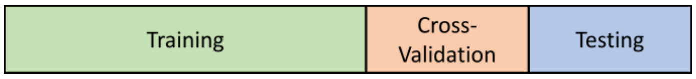


Figure 2.4: Train, test and cross-validation split of data.

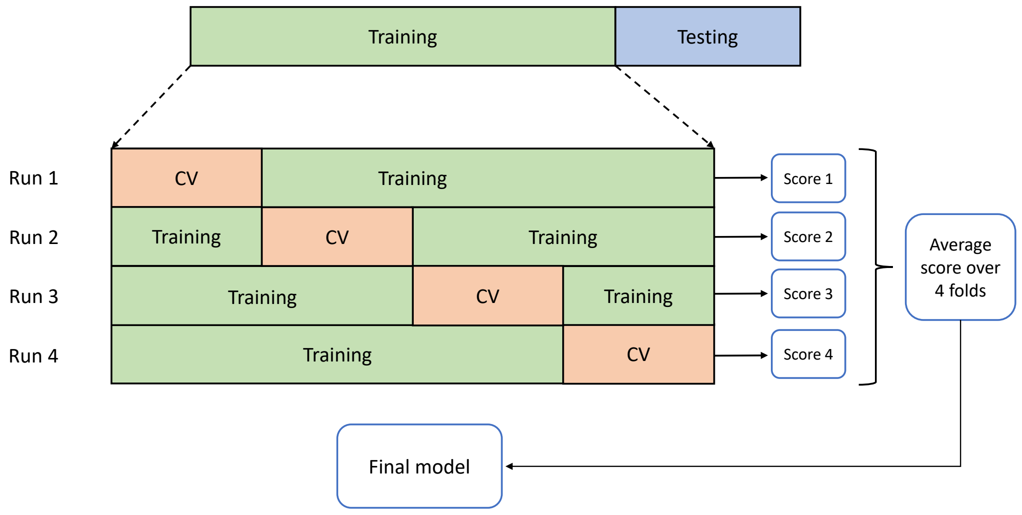


Figure 2.5: Example of a *k*-fold cross-validation using four splits.

2.3.2 Data Leakage

Data leakage occurs when information about the test data leaks into the training data. The algorithm adapts to this information and are then better prepared to predict the test data. This leads to an overly optimistic estimate of the performance since the algorithm has information about that of which it is trying to predict. Data leakage can occur at many stages in the development of a model. The most clear data leakage is when the raw data from the test set are used to train the algorithm. More hidden data leakage can occur during feature engineering. Usually scaling of the features are necessary for some classifiers to work. Scaling/standardisation are performed by calculating some scaling factor, for example the mean, based on the dataset. If the scaling factors are determined based on the entire dataset, data leakage would occur, since the scaling factors contain information about the distribution of the entire dataset. Thus, scaling must be done solely based on the training data and if a k-fold-CV splits are used, scaling should occur based on the training data within each fold.

The same is true for feature filtering methods. Feature filtering/selection are the procedure of eliminating uninformative features from the dataset. Again, if feature selection takes into account the entire dataset, then the remaining features are selected based on those who have the best description of the entire dataset, i.e. both the training and the test set. The algorithm will therefore perform better since the features are adjusted to fit the test data.

Another factor of data leakage are if the data, or features, in the training set and test set are to similar. This would be equivalent of using the same data for both training and testing. This was highly relevant for the dataset used in this thesis, since it contained a large number of identical samples. Thus extra care had to be taken during splitting of the data. It is enough if only a few of the features used contain information about the test set, as these features can quickly become dominant in the models.

Data leakage are easily identified by an unrealistically high performance. If the performance seem to good to be true, it should be investigated if that performance is a result of data leakage. Both in the literature and this thesis, data leakage and target leakage are used interchangeably, and essentially refers to the same phenomenon.

2.4 Classifiers

2.4.1 Support Vector Machine

Support vector machine (SVM) was developed in the 1990s and is a generalization of the maximal margin classifier [26]. A maximal margin classifier defines an optimal flat affine hyperplane based on training data to separate two classes. The hyperplane for a maximal margin classifier is in a subspace of one dimension lower than the original feature space, thus if the data is defined in a p-dimensional space, the hyperplane would be $(p-1)$ dimensions [26]. A 1-dimensional hyperplane separating two classes in a 2 dimensional feature space can be seen in fig. 2.6.

The margin in a maximal margin classifier determines the area in which the hyperplane is optimized. An optimal hyperplane is found by maximizing the distance to observations of different classes within the margin. The observations within the margin are called support vectors and are the only observations that affects the position of the hyperplane. With a larger margin, more support vectors influence the hyperplane and the classifier has a lower emphasis on observations on the wrong side of the decision boundary. This means a large margin are prone to high bias, while a small margin are prone to high variance.

The support vector machine classifier is an extension of the maximal margin classifier to non-separable, non-linear classes. With a non-linear decision boundary, a flat affine hyperplane of (p-1) dimensions are not suited for classification. By instead transforming the data set to a higher dimension, a linear decision boundary can be achieved. The support vector machine is developed based on the principle of enlarging the feature space to facilitate a non-linear decision boundary [24].

The SVM calculates the relationship between observations in a higher dimension through kernel functions. A kernel function is a mathematical function that quantifies the similarity of two observations, usually in an enlarged feature space. This relationship is then used to define the decision boundary of the classifier. This is equivalent to transforming the data to a higher dimension then defining a flat affine hyperplane to the transformed data. This way a non-linear decision boundary are achieved through the higher dimensional hyperplane. An important distinction is that with SVM the higher dimensional relationships are *calcu-*

lated through kernel functions, meaning the data is not *transformed* to a higher dimension. This trick, the kernel trick, reduces computations significantly and enables calculation of the infinite dimensional relationships.

The radial kernel (Radial Basis Function Kernel) is often used in support vector machines. The radial kernel calculates the relationship between observations in infinite dimensions and works similar to the weighted nearest neighbour model, where the closest observations has a larger influence on the classification of new observations [24]. The relationship calculated by the kernel function is the Euclidean distance between feature vectors of different observations in infinite dimensions. The radial kernel is expressed in eq. (2.2), where X, X' are two observations and γ is a positive constant [26].

$$K(X, X') = exp(-\gamma||X - X'||^2) \tag{2.2}$$

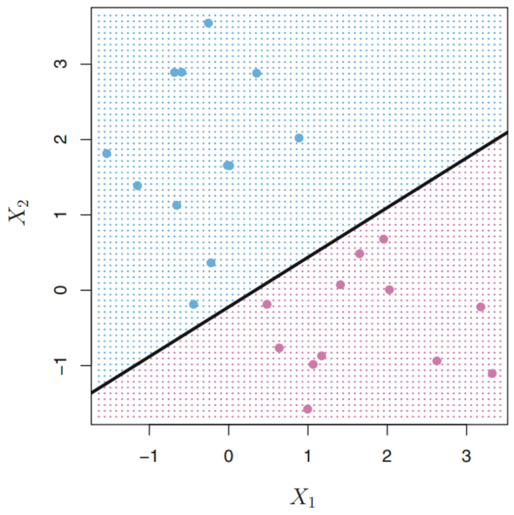


Figure 2.6: Example of a 1-dimensional hyperplane, black line, separating two classes, red and blue [26].

2.4.2 Logistic Regression

Logistic regression is a discriminative classifier that models the probability that an observation X belongs to a specific class Y by estimating $P(Y|X)$ [26]. This is done by fitting the logistic function to the training data set, thus performing regression through the logistic function. The logistic function is expressed in eq. (2.3a). The logistic function generalized for a logistic regression classifier is expressed in eq. (2.3b). The coefficients β_0 and β_1 are estimated based on the available training data through maximizing the likelihood function presented in eq. (2.3c) [24]. The likelihood function is maximized through gradient descent. The logistic regression classifier then classifies any new observation X_i by calculating the probability $P(Y|X_i)$ [26]. Hence, the output of the logistic regression classifier is the probability that an observation belongs to a certain class, i.e. a continuous variable.

$$\theta(x) = \frac{1}{1+e^{-x}} \tag{2.3a}$$

$$p(x) = \frac{e^{\beta_0+\beta_1 x}}{1+e^{\beta_0+\beta_1 x}} \tag{2.3b}$$

$$l(\beta_0,\beta_1) = \prod_{i:y_i=1} p(x) \prod_{i':y_i} (1-p(x)) \tag{2.3c}$$

The above equations are written for a single predictor and a Boolean output. However, logistic regression can be extended into multiple logistic regression to classify observations based on multiple predictors as well as multinomial logistic regression for any number of discrete output variables [26, 27]. For multiple predictors X_i one simply includes $\beta_i x_i$ in the eq. (2.3b) for each new predictor i.

Multinomial logistic regression are usually performed by using a one-VS-all classifier for each of the output variables. A one-VS-all classifier fits k classifiers for k output variables. Each of the classifier treats one of the classes as positive and the rest as negative. Predictions are then made based on the most confident classifier. If the classification problem consist of k classes, that is $Y = [y_1, y_2, ..., y_k]$, then the multinomial logistic regression classifier would have $k-1$ linear expressions in order to capture the distribution of each class. The distribution of the final class is simply found by $P(y_k) = 1 - P(y_{k-1})$.[26]

One problem often found in logistic regression with many parameters is the problem of overfitting/high variance. This can be solved by adding a regularisation term to the cost function. The regularisation term penalizes high values of the parameters, thus simplifying the cost function. By adjusting the weight of the regularisation term λ, one can tilt the algorithm in favor of either bias/variance. A large weight incentives smaller parameters, thus reducing the variance and increasing the bias. By utilizing the regularisation term, an optimal balance between bias and variance can be found for the respective algorithm and training set [24]. The cost function of logistic regression with regularisation is expressed in eq. (2.4), where θ_j represents the weight of predictor j.

$$J(\theta) = -\left[\frac{1}{m}\sum_{i=1}^{m} y_i log h_\theta(x_i + (1-y_i)log(1-h_\theta(x_i))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2 \tag{2.4}$$

2.4.3 K-Nearest Neighbour

The K-Nearest Neighbour classifier (KNN) is a supervised, non-parametric learning algorithm that compares new observations to the K-nearest observations in the training data. The class of the new observations are then determined as the most common class, or the *majority vote*, of its K-nearest neighbouring observations. When used in regression, the estimated $f(x_i)$ is the average value of the closest observations. For continuous variables, the nearest training observations are often determined by calculating the Euclidean distance. KNN for regression is expressed in eq. (2.5), where K is the number of neighbouring observations, N_0 denotes the neighbouring observations, y_i is the labels of the observations in N_0 and $\hat{f}(x_0)$ is the estimated value of $f(x_0)$ where x_0 denotes the prediction point [26].

The KNN classifier demands little computational power, since the training only consists of storing the feature vectors along with the target labels.

$$\hat{f}(x_0) = \frac{1}{K} \sum_{x_i \in N_o} y_i \tag{2.5}$$

2.4.4 Random Forest and Decision Trees

The random forest classifier utilizes decision trees in classification tasks. Decision trees are simple and intuitive classifiers that use stratifying or segmenting of the feature space for classification [28]. However, due to the simple design, decision trees are outperformed by more advanced supervised learning algorithms.

Decision tree classifiers segments the feature space into a set of regions based on the target classes [26]. The trees are split in a hierarchical manner, with the first splits containing a larger number of observations and branching out to smaller splits/nodes. A new observation will be given the class of the most commonly occurring class in the specific region/split which contains the observation. The division of the feature space is usually determined by either the Gini index or entropy, both seeking to maximize the node purity [26]. Node purity is a measure of how pure the split is in terms of classes. Maximum node purity occurs when a split contains only a single class. Decision trees usually performs well on a complex, non-linear relationship between features and target label. The hierarchical structure of decision trees are visualized in fig. 2.7.

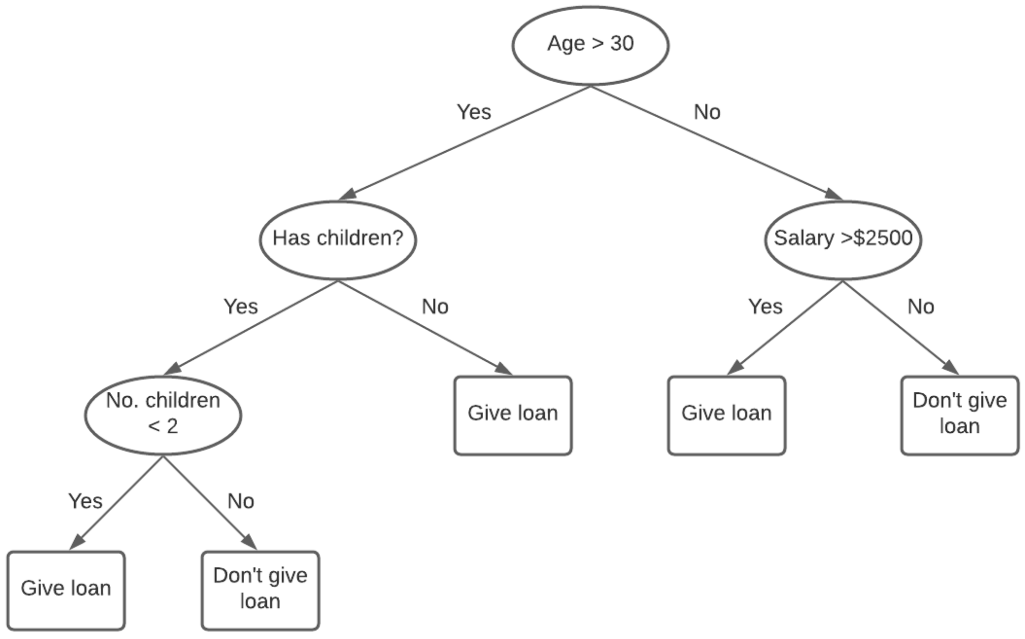


Figure 2.7: An example of a simple decision tree over wether or not a bank should give out loan.

The random forest classifier uses modified *bootstrap aggregation* (bagging) for reducing the variance compared to decision trees. Bootstrapping is used to create multiple training data sets from the original training data set, then multiple decision trees are built on the bootstrapped sample series to create a forest of trees. This process is called bagging, or bagged trees, and samples not included in the bootstrapped data set are called out-of-bag samples (OOB). In a classification task the majority vote of the bagged trees are used to classify new samples. Random forest decorrelates the bagged decision trees by choosing a random subset of predictors as split candidates each time a split in a decision tree is considered [26]. This is done to ensure sufficient reduction in variance at the existence of some strong predictors in the data set. Averaging highly correlated trees does not reduce variance significantly, thus by decorrelating the trees one ensures reduction in the variance of the classifier [26].

The random forest algorithm are well suited and often used for feature selection. This is done by evaluation of the OOB error and variable importance measures. The OOB error is the estimation error of the individual decision trees when tested on the OOB samples. The variable importance measure denotes the importance of each variable used at each split. The importance is determined either based on node purity, by means of the Gini variable importance measure, or the difference in predictive accuracy in the OOB samples when the individual variable is permuted [29]. The average importance of each variable across all decision trees in the RF are then evaluated and features deemed less important are excluded.

2.4.5 Boosted trees

Boosting is another form of improving the performance of decision tree classifiers. Opposed to bagging, boosting utilizes an ensemble of decision trees, where each tree contains information from the previously built trees and draws information from a modified version of the original data [26]. While bagged trees are deep to reduce bias, its often beneficial to have multiple shallow trees in a boosted classifier. The sequential trees in a boosted decision tree are formed on the residual from the previous decision tree. Each time a tree is formed, the training data is modified to contain information on misclassified observations, thus the seconding trees puts a higher emphasis on samples that are hard to classify. The trees are thus trained sequentially, making up an ensemble of trees each built on the information of the prior trees. This represents a slow learning algorithm, contrary to random forest where trees are uncorrelated and trained simultaneous. In machine learning, slow learning algorithms often outperform faster learners [26]. Depending on the method, the usual hyperparameters include; number of trees in the ensemble B; shrinkage parameter λ; interaction depth d [26]. Overfitting can be problematic if B is set to large, and a small d (shallow trees) are often beneficial.

Gradient boosted trees, like XGBoost, are some of the most popular decision tree based machine learning algorithms [30]. XGBoost has consistently placed among the top contenders in Kaggle competitions, with the winner of a 2015 challenge Owen Zhang stating "When in doubt, just use XGBoost" [8, 31, 32]. Explanation of the mathematics behind gradient boosted techniques are beyond the scope of this thesis, as most methods are available as open-source libraries developed by more competent computer engineers. XGBoost is both highly accurate and fast in classification tasks using a high-dimension feature space [33]. In [30] a method of reducing overfitting in boosted regression tree algorithm are proposed. By incorporating statistical significance, Welch's t-test, in tree construction, the generalizability of XGBoost improved significantly.

2.4.6 Artificial Neural Network

Artificial neural network is an AI technique that mimics the functionality of the human brain. The same way the brain transmits signals through and from neurons, ANN consists of a network of interconnected neurons or nodes. The individual neurons are organised in layers, with each neuron consisting of a weight and an activation function. The weights are determined during training of the algorithm. The activation function, number of layers and neurons are the hyperparameters which needs to be predetermined by the developers of the algorithm [24].

The activation function determines the complex relationship between variables and layers [34]. These activation functions produce non-linearity in the ANN. The performance and computation of the neural net depends heavily on the choice of activation function. The sigmoid function presented in eq. (2.3a) and ReLu (Rectified Linear Units) presented in eq. (2.6) are two examples of popular activation functions [34, 35, 36].

$$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases} \tag{2.6}$$

The Multilayer Perceptron (MLP) is a feed-forward neural network composed of multiple perceptrons. A perceptron is the simplest form of ANN and consist of only two layers: one input and one output layer. The perceptron are useful for classifying linear relationships, while the MLP can approximate non-linear functions [24, 27]. The layers in between the input- and output layer in an MLP are referred to as hidden layers. An MLP can have any number of layers and neurons, and the architecture has to be made to reflect the task at hand. For complex classification tasks a deep neural net with a large number of neurons might be beneficial, while less complex task can take advantage of a shallower net with fewer neurons. Error back-propagation are typically used when training an MLP. This method calculates the classification error at each neuron and adjusts the weights to minimize this error. The error is propagated from back to forth in the ANN architecture over a series of epochs, until a stoppage criteria occurs. At each epoch the weights of the neurons are adjusted to improve the predictive performance. Error back-propagation requires a at least piecewise differentiable activation function since the error is calculated through the gradient of the cost function [24].

A simple MLP with one hidden layer and three neurons is displayed in fig. 2.8. The x_0 neuron is referred to as the bias unit and serves as a fixed input [24]. The bias, along with the weights, are fitted during training. Examples of more advanced types of ANN are the radial basis function neural net (RBFNN), convolutional neural net (CNN) and recurrent neural net (RNN). CNN are often used in image recognition and RNN are a natural choice for text recognition.

2.4.7 Ensemble Learners

Ensemble learners are defined as a group of individual learning algorithms trained cooperatively on a data set to solve a learning problem. The combination of multiple individual learning algorithms are performed to enhance prediction performance beyond what any of the underlying learning algorithms could obtain alone. There are multiple techniques used to create ensemble learners, some of which are discussed in previous sections. Among the most common methods are *baggin, boosting, stacking* and *Bayesian* based methods (Bayes

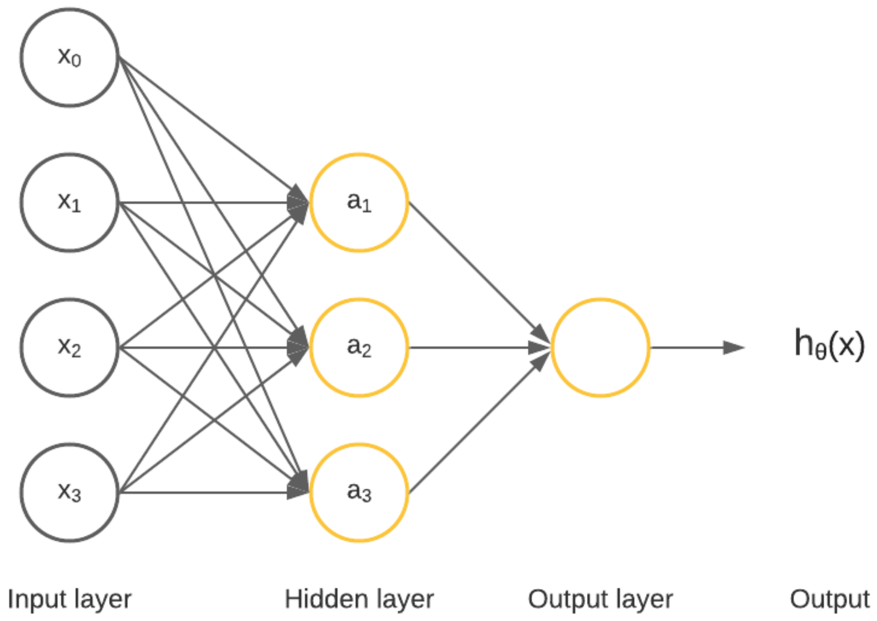


Figure 2.8: Example of a simple multilayer perceptron.

optimal classifier, Bayesian model averaging and Bayesian model combination). Both RF and XGBoost are forms of homogeneous ensemble classifiers, as the base classifiers are of the same type. XGBoost has also been used in ensemble with neural networks to boost performance [37].

A stacking classifier combines several different base classifiers and one *meta-classifier*. The constraint of using different classifiers ensures diversity and independence in the base classifiers. The base classifiers are independent both in training and classification. The meta-classifier serves to combine the output of the base classifiers such that the prediction accuracy is maximized [38]. The stacking classifier has a hierarchical structure with the output from the base-classifiers serving as input for the meta-classifier. For a stacking classifier to improve performance all base classifiers must have and accuracy better than random chance [38]. In [8] a stacking classifier was used to improve performance in detection of ITSC in SPSGs.

2.5 Imbalanced Data Set

The balance of a dataset refers to the ratio of samples from each of the embedded classes. An imbalanced dataset consists of one or several minority classes of which there are significantly fewer samples. This can cause major problems in the machine learning, manly that there are too few samples for the training of the algorithm such that the predictive performance on the minority class suffers. When faced with an imbalanced dataset the classifiers tend to favour the majority class when determining weights and decision boundaries. The amount of influence of imbalance is dependant on the classifier design.

One way of dealing with an imbalanced dataset is through under-sampling of the majority class. Under-sampling means choosing a subset of the majority class in such a manner that the balance is restored. There are manly two types of under-sampling: prototype selection and prototype generation. In prototype selection algorithms, the subset is chosen from the original dataset such that the new set S' is defined by $|S'| < |S|$ and $S' \in S$ [39]. The simplest form of prototype selection under-sampling is choosing a random subset of the majority class. More advanced methods incorporate classifiers in selection of the subset. Edited Nearest Neighbour from the imbalanced learn module in python applies a nearest neighbour algorithm which edits the dataset by removing samples that does not agree enough with the neighbourhood [39]. This filters outliers and noisy samples such that the resulting subset contains samples most representative of the "neighbourhoods". A pitfall with this method is one cannot know if the samples excluded through the algorithm represents noise or factual samples. As such, relevant samples might be filtered out resulting in a biased dataset. Prototype generation generates new samples based on the original dataset such that $|S'| < |S|$ and $S' \notin S$. The new samples are synthesized based on the samples in the original dataset. Prototype generation under-sampling has not been utilized in this thesis.

Over-sampling methods generates new samples of the minority class to adjusts the balance of classes in the training data. As with prototype generation, these new synthetic samples are generated based on the samples in original dataset. A naive approach to over-sampling is through over-sampling by replication, i.e. duplicating samples of the minority class. Duplicating samples has been shown to not significantly improve minority class recognition and often leads to overfitting [40]. A popular technique of over-sampling is the Synthetic Minority Oversampling Technique (SMOTE) [39, 40]. SMOTE create synthetic samples from the minority class by a k-nearest neighbour approach. For each of the samples in the minority class, the k-nearest neighbours of that class are considered and the difference between the feature vectors of the respective samples is evaluated. This difference is then multiplied with a random number between 0 and 1 which is stored as a synthetic sample in the training set. The new sample is thus a random sample on a line segment intersecting the nearest neighbours in the feature space. According to [40], this forces a more general decision region for the minority class, which increases the generalizability of the classifier. The amount of over-sampling is determined based on the k in k-nearest neighbours. With $k = 2$ the minority class is over-sampled by 200%, meaning doubling the amount of samples of the minority class. SMOTE is sensitive to outliers and noisy samples, as it will generate synthetic samples in these regions as well, effectively increasing the amount of noisy samples in the training data. This behaviour is illustrated in fig. 2.9.

An important distinction for both over-/ and under-sampling is that only the training data should be adjusted. Putting the test data under the loop could easily result in an over-optimistic evaluation of the algorithms performance. In addition, an imbalanced test-set provides no problems for the design/training of the algorithm, it only influences the ability to estimate the performance of the respective algorithm. As with all work within artificial intelligence, a test set should be extracted at the beginning and stored away only to be used for evaluating the performance of the finalised algorithm.

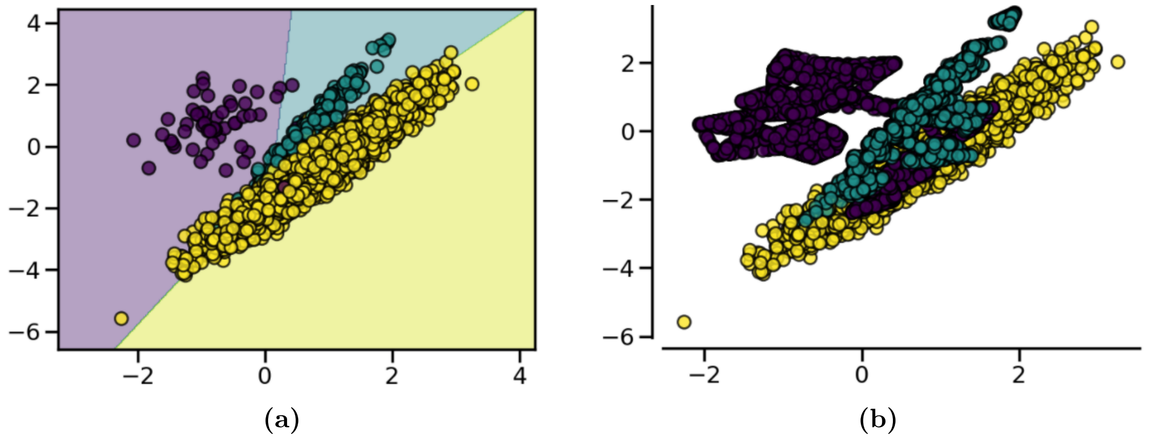


Figure 2.9: Illustration of SMOTE behavior at the presence of outliers/inliers: (a) Original data set, (b) dataset after resampling with SMOTE. Different colours represents the different classes [39].

2.6 Assessment Criteria

For the purpose of readability the following terms are used throughout this thesis when discussing the performance and performance parameters of machine learning algorithms:

- True positive (TP): Correctly classified as the positive class
- True negative (TN): Correctly classified as the negative class
- False positive (FP): Falsely classified as the positive class
- False negative (FN): Falsely classified as the negative class

Prediction accuracy is defined as the ratio of correctly classified observations to all classifications, as shown in eq. (2.7a). This is a simple form of evaluating the overall performance, however, it might might give misleading results in the presence of imbalanced data. If for example, a data set contained 95% class A and 5% class B, then a classifier that answers class A regardless of input would have an accuracy of 95%. This would however be a very bad classifier, contrary to the high accuracy.

Precision is used to quantify the classifiers ability to correctly classify observations as belonging to a class. High precision gives confidence in that any positive classification corresponds to a true positive observation. It is defined as the ratio of true positives to all positively classified observations, as shown in eq. (2.7b) [25].

Recall, sometimes referred to as sensitivity, describes the probability that an observation is correctly classified as belonging to the class. It gives information about how well the algorithm detects the class, and a high recall gives confidence in that all positive samples are detected. Recall is defined as the ratio of true positives to the total number of positive classes in the test data, as shown in eq. (2.7c) [25].

Specificity is the true negative rate, and informs about the rate of true negatives to all negative classes in the test data, as shown in eq. (2.7d) [25]. It is the equivalent of recall for negative cases.

$$accuracy = \frac{TP + TN}{\text{all classifications}} \tag{2.7a}$$

$$precision = \frac{TP}{TP + FP} \tag{2.7b}$$

$$recall = \frac{TP}{TP + FN} \tag{2.7c}$$

$$specificity = \frac{TN}{TN + FP} \tag{2.7d}$$

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{2.7e}$$

The $F_1 score$, shown in eq. (2.7e), describes the harmonic mean between precision and recall [41]. The $F_1 score$ is derived from the weighted $Fscore$ used in statistics, with precision and recall evenly weighted. One drawback of using the $F_1 score$ as an evaluation metric is that the true negatives are ignored. In classification problems it can be of equal or greater importance to correctly classify negative samples. In addition, recall and precision are evenly weighted. Depending on the problem at hand, either recall or precision might induce a heavy cost, and should thus be weighted accordingly. Take for example the classification problem of detecting a rare but fatal disease. In this case the recall should be weighted higher, as it is more important to identify all those who have the disease than to reduce the false positive rate. With a higher weighted recall one would have more people falsely identified as having the disease, but all those who have the disease would more likely get detected, thus saving more lives. However, if the treatment of the disease is costly or dangerous then there might be incentive to reduce the false positive rates, i.e. higher weighted precision.

Another evaluation metric often used in machine learning is the Area under the Receiver Operating Characteristics curve (ROC AUC). The Receiver Operating Characteristics describes the ratio of true positive rate (TPR) against the false positive rates (FPR) at different classification threshold. The classification threshold determines what is required for a sample to be classified as the positive class. In terms of the mentioned metrics, this means plotting the recall versus (1-specificity) [42]. The usual practise when comparing classifiers is to evaluate the area under the curve (AUC). A perfect classifier has a ROC AUC of 1, while a classifier no better than random chance has a ROC AUC of 0.5. The ROC AUC has been proven theoretically and empirically to outperform the accuracy metric [41]. An example of the ROC curve is displayed in fig. 2.10. The graph is constructed by evaluating the TPR and FPR at different thresholds. As the classification threshold decrease, more samples will be classified as positive leading an increase in both the True Positives and the False positives. The points $(0, 0)$ and $(1, 1)$ represents zero positively classified samples and all samples classified as positive, respectively.

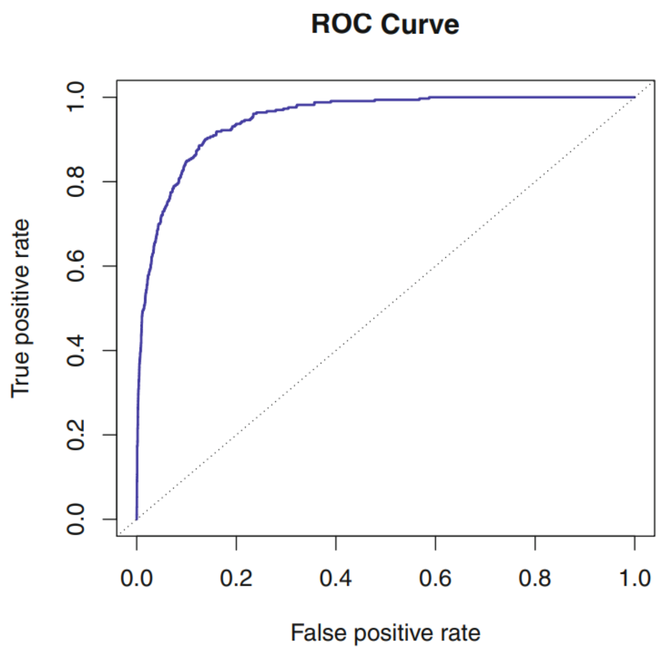


Figure 2.10: Example of the ROC curve [26]

.

2.7 Signal Processing for Feature Extraction

2.7.1 Fourier Transform

The second paragraph of this subsection is adapted from [9].

The Fourier transform (FT) reveals the frequency content of a time series by transforming the series from the time domain into the frequency domain. The Fast Fourier Transform (FFT) is a method developed to compute the Discrete Fourier Transform (DFT) in a computational efficient manner, and is the most widely used method of computing the Fourier transform. The FT is restricted to stationary signals.

Artefacts, such as spectral leakage and aliasing, serves as one of the major problems in employing FFT on real life signals. Artefacts can cause errors in the magnitudes and side-lobe distortion, which can lead to misinterpretation of results [43]. Artefacts arise from when the FFT is taken on a range that does not correspond to the period such that non-zero values are developed to non-existent frequency components.

In [9] the resampling method developed by prof. Bruce Mork was used to eliminate artefacts and spectral leakage to obtain an accurate FFT spectrum. It was concluded that the adjustment of the fundamental frequency due to slight speed deviations in the generator gave the largest impact on the spectrum, and resulted in the most accurate depiction of the frequency content of the signal. It was based on this deemed a superior method to that of FFT combined with windowing functions. However, locating the correct fundamental frequency requires a trial-and-error based approach to the analysis of the FFT spectrum of all signals in the database. As the machine learning algorithm is dependant on a considerable quantity of samples for both training, cross-validation and testing, the resampling method was deemed unusable on a project with a large dataset of signals with a reasonable time-schedule. As such, this thesis utilizes the well known FFT with windowing function to extract frequency content.

2.7.2 Wavelet Transform

The wavelet transform is a method of extracting the time-frequency resolution from a time-series. While FFT transforms from the time domain into the frequency domain, the wavelet transform maintains the time information and maps frequencies with respect to time. This is valuable information that can tell researchers about time sensitive information like discontinuity, singularity or abrupt changes. The wavelet transform performs well on non-stationary signals.

The fundamental principle of the wavelet transform is to move a *mother wavelet* along the signal and extracting the wavelet-signal correlation. This is done through a convolution of the wavelet and the raw signal [44]. A wavelet is a wave-like oscillation with zero mean and finite duration, with the mother wavelet denoting the shape of the chosen wavelet. Different mother wavelets are presented in fig. 2.11. The choice of wavelet function should reflect the raw signal and different wavelet function often give very different results. A wavelet is expressed mathematically through eq. (2.8a), where a, b are the scaling and shifting parameters, respectively, and ψ is the mother wavelet function [44]. The scaling parameter stretches or compresses the wavelet. A large scale factor, meaning a stretched wavelet, is useful for catching low frequency components and vise versa. The shifting parameter delays or advances the onset of the wavelet along the signal.

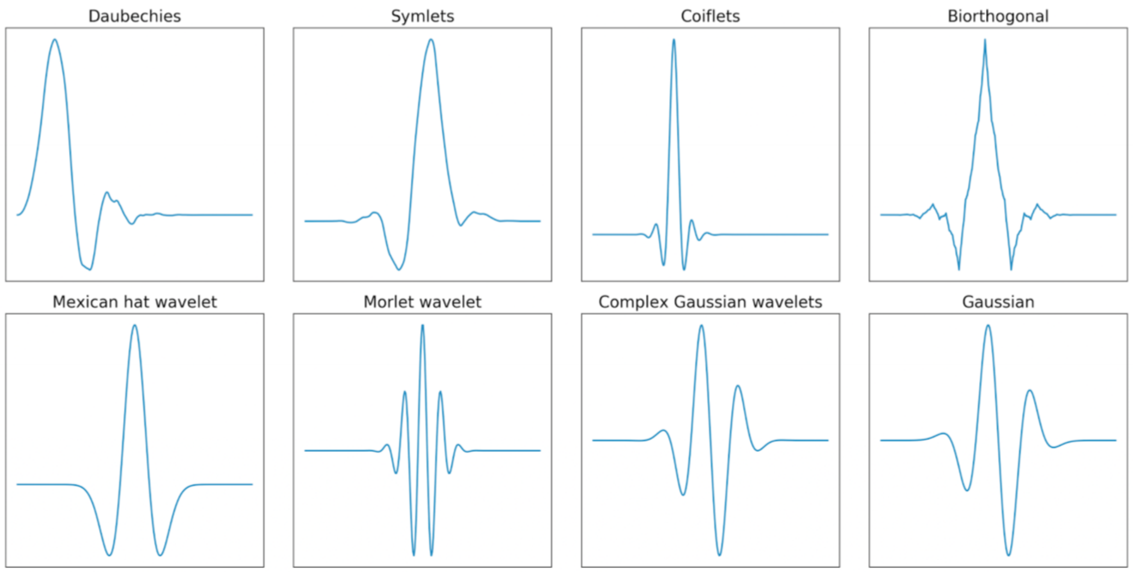


Figure 2.11: Examples of different wavelets [44].

The Continuous Wavelet Transform (CWT) is a computational expensive method of implementing the wavelet transform on a signal. It performs the wavelet transform for all possible scale factors and shifts the wavelet by an infinitesimal length along the signal. As such, the CWT obtains very high time-frequency resolutions and are often displayed in a scalogram. The CWT is expressed in eq. (2.8b) where $x(t)$ is the signal to be analysed [44, 45].

The alternative method to compute the wavelet transform is the Discrete Wavelet Transform (DWT), expressed in eq. (2.8c) [44]. The DWT gathers sufficient time-frequency information at a significantly lower computational cost. The DWT puts the signal through a series of high-pass and low-pass filters based on the mother wavelet. This decomposes the signal into a set of frequency ranges until a pre-defined decomposition level is reached. At each decomposition level the high-pass filtered signal and the low-pass filtered signal is stored as detail- and approximation coefficients, respectively. The the low-passed signal then goes through to the next decomposition and the process is repeated. The DWT method is illustrated in fig. 2.12, where D_i and A_i represents the detail- and approximation coefficients of layer i. Similar to the EMD, the DWT results in a set of sub-bands of a certain frequency range As with the CWT, the DWT may be visualized through a scalogram. At each decomposition level half the frequencies in the decomposed signal is extracted. This means half the signal is eliminated at each step according to the Nyquist rule.

$$\psi_{a,b}(t) = \frac{1}{\sqrt{|a|}}\psi\left(\frac{t-b}{a}\right) \tag{2.8a}$$

$$CWT(a,b) = \int_{-\infty}^{\infty} x(t)\frac{1}{\sqrt{|a|}}\psi\left(\frac{t-b}{a}\right)dt \tag{2.8b}$$

$$DWT(j,k) = \int_{-\infty}^{\infty} x(t)\frac{1}{\sqrt{2^j}}\psi\left(\frac{t-2^jk}{2^j}\right)dt \tag{2.8c}$$

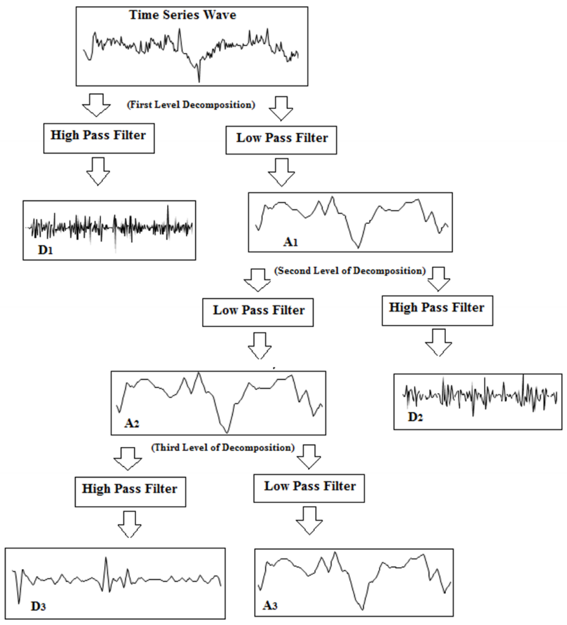


Figure 2.12: The DWT process [44].

The energy contents of the wavelet decompositions have been used successfully in fault detection [8]. In signal processing a number of methods have been developed to extract energy contents based on *energy operators*. In [8] it was found that Instantaneous Wavelet Energy (IWE), Teager Wavelet Energy (TWE), Hierarchical Wavelet Energy and Relative Wavelet Energy (RWE) has a strong correlation to ITSC.

IWE is calculated based on the conventional energy calculation used in signal processing by taking the area under the square magnitude of the wavelet coefficients (eq. (2.9a)).

The TWE uses the Teager energy operator to calculate the energy content of the wavelet decomposition, as shown in eq. (2.9b) [8, 46]. This method is found to be more robust to noise.

The HWE diminishes end-effects by taking the energy of the middle part of the DWT coefficients [8]. The HWE are expressed in eq. (2.9c).

The RWE is calculated in terms of the wavelet energy of each decomposition level relative to the total energy of all levels [8, 45, 47].

For eq. (2.9a) to eq. (2.9d), $w_j(r)$ is the wavelet coefficients of level j and $r = 1, ..., N_j$ with N_j being the number of coefficients in the decomposition level [8]. N_J is the number of coefficients in the previous decomposition level. The equations are written for the python implementation of DWT, where the decomposition levels are stored from back to forth, i.e from low frequency to high frequency. This means that N_j is greater than N_J

$$IWE_j = log_{10}\left(\frac{1}{N_j}\sum_{r=1}^{N_j}(w_j(r))^2\right) \tag{2.9a}$$

$$TWE_j = log_{10}\left(\frac{1}{N_j}\sum_{r=2}^{N_j-1}|(w_j(r))^2 - w_j(r-1)\cdot w_j(r+1)|\right) \tag{2.9b}$$

$$HWE_j = log_{10}\left(\frac{1}{N_j}\sum_{r=\frac{N_j-N_J}{2}}^{\frac{N_j+N_J}{2}}(w_j(r))^2\right) \tag{2.9c}$$

$$RWE_j = \frac{E_j}{E_{total}} \quad , \quad E_j = \sum_{r=1}^{N_j}(w_j(r))^2 \quad , \quad E_{total} = \sum E_j \tag{2.9d}$$

2.7.3 TSFRESH

Time Series Feature Extraction Based on Scalable Hypothesis Tests (TSFRESH) is a feature extraction package developed for python [48]. The package supports up to 778 features describing meta-information about a time series. Most feature are statistical measures, such as max/min/mean values, autocorrelation, standard deviation, kurtosis, skewness, entropy etc. For a full list of features included in the TSFRESH python package see [48]. As reported in [8], the features extracted through TSFRESH paralleled that of the DWT energies and outperformed the FFT features, implying that TSFRESH is a valuable feature extraction tool for fault detection of electrical machines.

Furthermore, the package can be used to filter out irrelevant features based on the scalable hypothesis test. Feature filtering/selection are done by calculating the p-values between each feature and the target value, thus quantifying their significance in predicting the target. The method of evaluating the p-values of the features are the Benjamini-Yekutieli procedure, which can be studied in detail in [49]. In short, a feature is deemed relevant if it is not statistically independent from the target prediction, and features with a p-value below a certain threshold are rejected through the selection process. The TSFRESH procedure including feature selection are illustrated in fig. 2.13

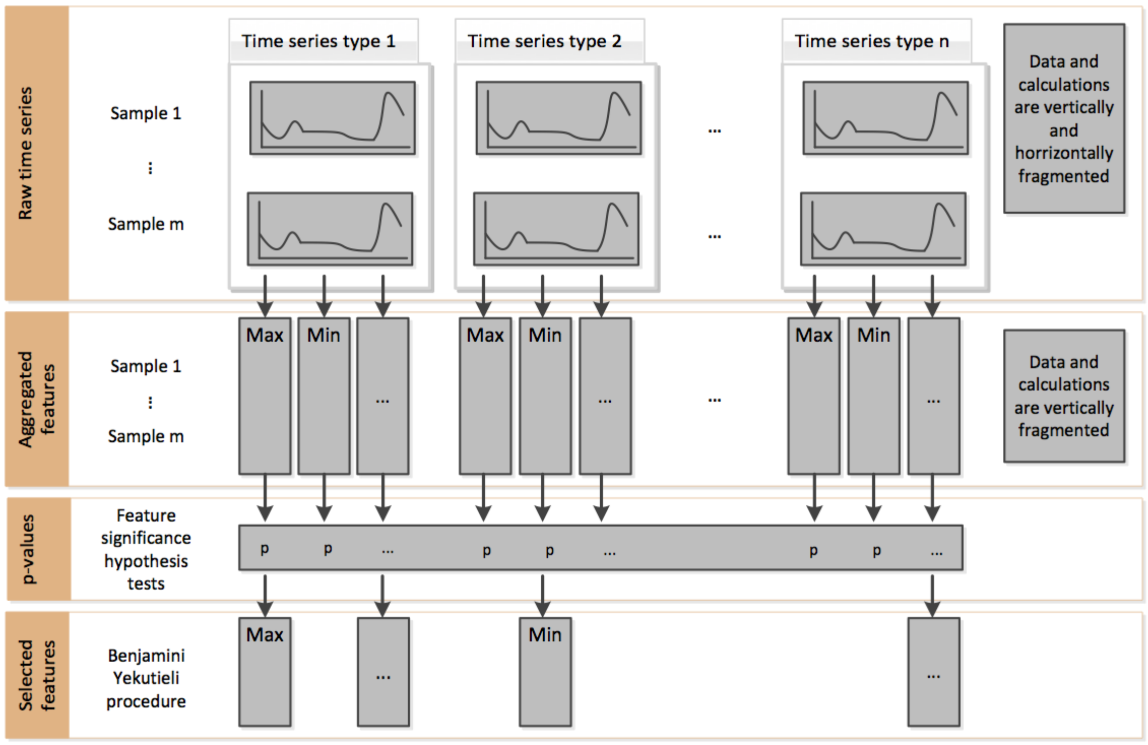

Raw time series
Time series type 1
Time series type 2
Time series type n
Sample 1
⋮
Sample m
...
Data and calculations are vertically and horrizontally fragmented
Aggregated features
Sample 1
⋮
Sample m
Max
Min
...
Data and calculations are vertically fragmented
p-values
Feature significance hypothesis tests
p
Selected features
Benjamini Yekutieli procedure

Figure 2.13: Illustrative explanation of the TSFRESH algorithm [48].

Chapter 3

Laboratory Setup

This section explain the laboratory setup used to obtain the measurements used in the machine learning algorithm. The state of art and related work were reviewed, and an identification of the relevant background material were carried out in the project preceding this thesis [9]. The presentation from the project report is included in the following sections, with an added section regarding the loading of the machine.

3.1 Generator Specifications

The laboratory generator is a 14-pole 100kVA synchronous generator located in the NTNU National Smart Grid Laboratory. The generator is made to resemble a typical hydropower generator, only scaled down in size for practical reasons. The rated speed of the machine is 428.6rpm with a fundamental frequency of 50Hz. The rated values and specifications of the laboratory generator are summarised in table 3.1 and table 3.2.

The generator is custom made to investigate faults in synchronous generators and can be imposed with broken damper, static eccentricity and inter-turn short-circuit faults of varying degree of severity. Broken damper bar are imposed by removing the respective damper bars from the pole. Static eccentricity are created by moving the stator frame such that the rotor is displaced with respect to the stator origin. By adjusting the external connections of the rotor windings the generator can be imposed with up to 30% ITSC per pole in two of the poles of the machine.

The generator is a fractional slot machine, with $q = 2 + 5/7$ as found in eq. (3.1). The pole pitch and coil pitch can be calculated from eq. (3.2), where α_{cp} is the coil-pole fraction, θ_p is the angular pole pitch, R_{si} is the inner stator radius and τ_p, τ_c are the pole and coil pitch [50]. The pole pitch and coil pitch are found to be approximately 161mm and 119mm respectively, which means the windings of the laboratory generator are short-pitched.

$$q = \frac{N_s}{N_{ph}N_p} = \frac{114}{3 * 14} = 2 + \frac{5}{7} \tag{3.1}$$

$$\tau_p = R_{si}\theta_p \tag{3.2a}$$

$$\tau_c = \alpha_{cp}\tau_p \tag{3.2b}$$

Table 3.1: Rated values of the laboratory generator

Nameplate values	
Nominal power	100 kVA
Nominal voltage	400 V
Nominal current	144.3 A
Nominal speed	428.6 rpm
Nominal frequency	50 Hz
Nominal power factor	0.90
Nominal exc. current	103 A
No-load exc. current	53.2 A

Table 3.2: Specifications of the laboratory generator

Specifications	
Number of poles	14
Number of slots	114
Damper bars per pole	7
Winding connection	Star
Turn per field winding	35
Outer rotor diameter	646.5 mm
Outer stator diameter	780 mm
Nominal air-gap length	1.75 mm

3.2 Laboratory Setup

The generator was powered by a 90kW induction machine with 4 poles and a rated speed of 1482rpm. The induction machine was connected to the generator through a gearbox. A programmable converter was used to supply the induction machine. The induction machine could be overloaded to supply the rated power of the generator, however this was not utilized during this project.

The output power of the generator was supplied to a local grid at NTNU. The load was then controlled by adjusting the input power to the induction motor and the excitation current of the generator supplied by an external converter. The speed of the machine was kept constant by the synchronization of the generator to the grid. Due to the controllers and grid influence, neither the input power to the induction machine, the excitation current nor the load drawn by the local grid could be kept constant during the experiments. Thus, for each of the measurements series the loading of the generator varied slightly. It was found that during some of the runs, the magnitude of the power variations reached about 20%, with the lowest load having the largest fluctuations. The fluctuations in load was limited by sampling the signals at near stationary conditions. However, this was not always possible, thus the load varies to some degree within each measurement series. This was not considered a major problem since the load of an industrial hydropower generator would also be varying. The load cases are presented in table 3.3. The values presented in table 3.3 are all approximated values due to the varying load.

Table 3.3: Load cases

Load type	1	2	3	4	5	6
Active Power	22-28kW	45-50 kW	50-60kW	70-78 kW	80-85 kW	85 kW
Reactive Power	7-10kVar	1-2 kVar	12-17 kVar	20-24 kVar	26-30 kVar	30 kVar
% of full-load	~ 30%	~ 50%	~ 60%	~ 75%	~ 85%	~ 90%

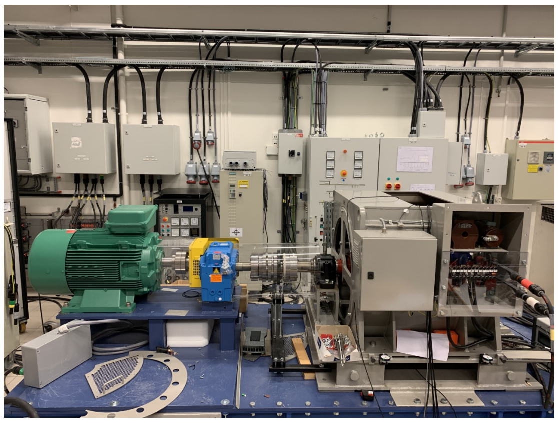
The laboratory setup is shown in fig. 3.1, with the induction machine used as prime mover visible in fig. 3.1a. Figure 3.1b shows the generator with the rotor windings on display. The field windings were shorted by fastening a copper plate between the respective windings seen in fig. 3.1b. The generator was imposed with ITSC in one pole according to table 3.4.

Table 3.4: Percentage of shorted turns in the experimental measurements.

	Healthy	1 Turn SC	2 Turn SC	3 Turn SC	7 Turn SC	10 Turn SC
% of shorted turns in pole	0%	2.9%	5.7%	8.6%	20%	29%
% of total turns	0%	0.002%	0.004%	0.006%	0.014%	0.02%

3.3 Measurements

As part of the work of this thesis, new measurements where taken of the laboratory generator. This was a necessity for further developing of the machine learning algorithm in [8]. For the vibration signal two piezoelectric accelerometers was attached to the stator core in three different locations. For capturing the stray field, three identical search coils developed by Hossein Ehya were placed on three different locations on the stator core [51]. The sensors were placed on top of the machine for detection of ITSC and on the side of the machine for detection of SE. The location was then varied in the axial direction such that the radial position of the sensors stayed the same for each fault. Thus the measurement series consist of measurements from three different axial positions. The sensor locations used and the sensors are shown in fig. 3.2. As seen in fig. 3.2, the sensor locations are numbered from 1-3, and will henceforth be referred to by those numbers.. Voltage and current was sampled


(a)

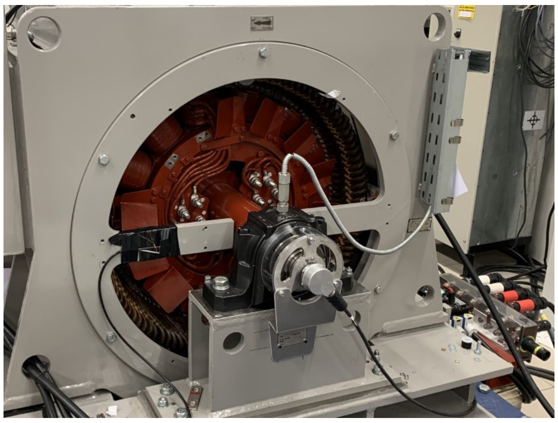
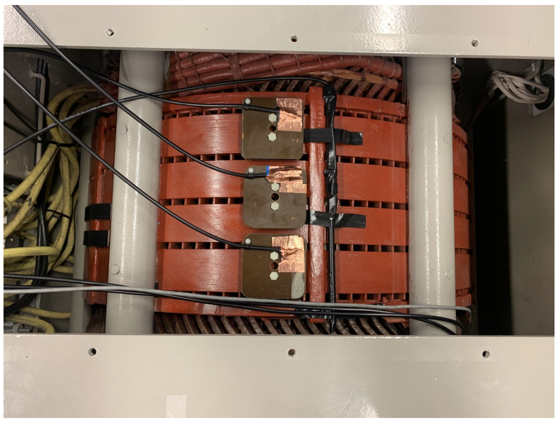
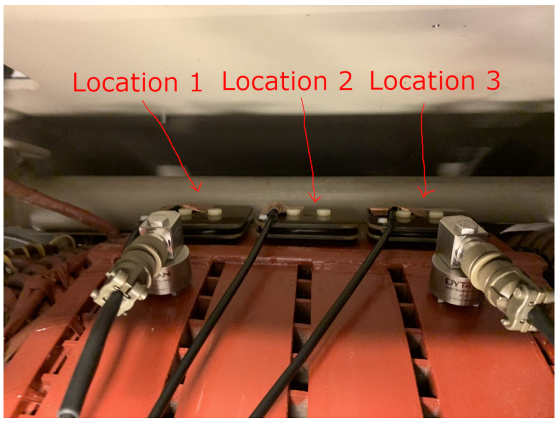

(b)

Figure 3.1: The laboratory setup.

using a voltmeter and an AC current clamp meter, respectively. All signals was sampled for 40 seconds using a sampling frequency of 10kHz.


(a)



(b)

Figure 3.2: Sensor locations: (a) location used for detection of ITSC, (b) location used for detection of SE.

An early problem discovered is the inherent imbalanced state of the dataset. Each case was measured once for every load, meaning by default the dataset ratio of faulty to healthy was 1/6 for ITSC and 1/3 for SE. It was based on this decided to measure three locations as well as one additional loading for the healthy case to facilitate balancing the dataset by means of under-/over-sampling. The different sensor locations and loadings were also chosen to diversify and increase the dataset. The ratios of healthy and faulty before adjusting the datasets are: 19% healthy, 81% faulty for ITSC and 37% healthy, 63% faulty for SE. A summary of the measurements for vibration and stray field are expressed in table 3.5. Full list of all the measurements can be found in appendix A.1

Stator current and voltage was measured for the load cases and fault cases expressed in table 3.3 and table 3.4, as well as for 10% and 20% SE. The initial idea was to perform sensor fusion of vibration, stray field, airgap field, current and voltage. However, the airgap field sensors were damaged, thus new measurements could not be conducted. Based on this, airgap magnetic field was decided to be eliminated from the project, as it would put a high constraint on the dataset. By using current and voltage in sensor fusion, the size of the dataset were limited to the sampled signals of current and voltage. This means that for stray and vibration, only one sensor location could be utilized in combination with current and voltage, since the sensor location could not be varied for the latter. To further clarify, in order to have sensor fusion, there must be signals to fuse, i.e. there must be an identical number of samples for each of the signals. Due to the limitations posed by current and voltage, two scenarios of sensor fusion were made. The scenarios are expressed in table 3.6. No-load measurements was excluded from scenario 2, since the no-load current is always zero.

Table 3.5: Summary of measurements of vibration and the stray field.

Case	# Sensor locations	Load type	Total number of samples
Healthy	3	NL, L1, L2, L3, L4, L5, L6	21
1 turn SC	3	NL, L1, L2, L3, L4, L5	18
2 turn SC	3	NL, L1, L2, L3, L4, L5	18
3 turn SC	3	NL, L1, L2, L3, L4, L5	18
7 turn SC	3	NL, L1, L2, L3, L4, L5	18
10 turn SC	3	NL, L1, L2, L3, L4, L5	18
10% SE	3	NL, L1, L2, L3, L4, L5	18
20% SE	3	NL, L1, L2, L3, L4, L5	18
Total			147

Table 3.6: Sensor fusion scenarios.

Scenario	Fused signals	Measurements included
1	Stray magnetic field Vibration	3 sensor locations: NL, L1-L6
2	Stray magnetic field Vibration Stator current Stator voltage	1 sensor location: L1-L6

Chapter 4

Methodology and Results

This chapter contains the methodology and results of the thesis. The layout of the chapter is hierarchical, following the procedure of building a machine learning model, with each method and its accompanying results described in its respective section. The methods used in this chapter was based on an adaption of the methodology used in [8]. With the addition of signal fusion and due to the signals used in this thesis differing significantly from those used in [8], the results from the preceding thesis could not be used directly. It was deemed a necessity to perform the full procedure of building a machine learning model anew. This includes exploratory data analysis, classifier and feature selection, and optimization of hyperparameters. Following is the chapter outline:

Section 4.1 - Pre-Processing: Description of the pre-processing performed on the measurements along with the method of signal fusion.

Section 4.2 - Feature Extraction: Description of the implemented feature extraction methods accompanied with the results from feature extraction.

Section 4.3 - Exploratory Data Analysis: Evaluation of the features extracted from the measured samples.

Section 4.4 - Feature Filtering: Implementation and results from feature filtering through random forest and TSFRESH.

Section 4.5 - Algorithm Selection and Data Set Performance: Description of the initial classifiers and their results on the different datasets through a 5-fold-CV split.

Section 4.6 - Hyperparameter Selection: Optimisation of hyperparameters through grid-search and the results from the optimised classifiers.

Section 4.7 - Performance of algorithms on hold-out data: The optimized classifiers are tested on the hold-out-data.

Section 4.8 - Improved Performance: - This section incorporates a new splitting strategy and displays the weaknesses with the previously implemented methodology.

Section 4.9.1 - Feature Importance: The 20 most relevant features for Logistic regression and XGBoost are displayed.

Section 4.10 - Generalizability: Evaluation of generalizability of classifier on samples from industrial hydropower generators.

4.1 Pre-Processing

The goal of the machine learning model was to distinguish between a healthy and a faulty machine. A faulty machine is defined as a machine suffering from any degree of either ITSC or SE. Based on this it was decided to make a Boolean classifier, with healthy case represented by a Boolean False and faulty case a Boolean True. The individual measurements was therefore labeled according to the state of the machine from which it was extracted. All samples measured during a fault, irrespective of fault severity, were assigned a Boolean false as label.

As previously mentioned, machine learning is dependant on a large amount of data to obtain a high performance. Due to the limited number of samples obtained from the laboratory generator it was decided to split each sample into several reduced sample series to increase the training data available for the algorithm. In [8] a similar split was performed, by splitting the signal at a rising zero-crossing. Due to the nature of both stray flux and vibration this method was not practically feasible since the signals oscillate around zero-crossings. It was instead decided to split the signals by a fixed number of mechanical revolutions as this would ensure samples of equal length and with the fault signature contained in each signal. Splitting by a fixed number of data points also keeps the periodic nature of the signal. One mechanical revolution was chosen as split sequence, since this is the minimal signal length while maintaining the fault signature in each reduced sample series. This resulted in 250 reduced sample series per measurement. In addition, one electrical period was skipped at each split to ensure that the fault signature appeared in every possible position in the samples. This would also cause dissimilarities in the samples, making the data set more representative. The reduced sample series are henceforth referred to as RSS, while the original measurements are referred to as OSS (original sample series).

An analysis of the RSSs revealed that the RSSs obtained from the same OSS were not differentiable. They were therefore assigned an id, referred to as the *OSSid*, representing the OSS it was drawn from. Having RSS with the same id in both the train set and the test set would be equivalent of training and testing the machine learning algorithms on the same data, thus creating target leakage. Initially every OSS was assigned a unique id, which was shared by the RSSs obtained from that specific OSS.

Both DWT and FFT are dependant on signal length. FFT has been found to be ineffective on signals shorter than 8 mechanical revolutions [9, 51], and in [8] it as found that 4 mechanical revolutions was the minimal signal length to obtain the DWT energies. This puts a heavy constraint on the maximum number of RSS. To avoid this constraint the OSS was first split into RSS of one mechanical revolution, then the individual RSSs was concatenated before feature extraction. Thus each of the RSSs had sufficient length for feature extraction while maintaining the maximum number of samples in the data set. The splitting of the OSS are illustrated in fig. 4.1

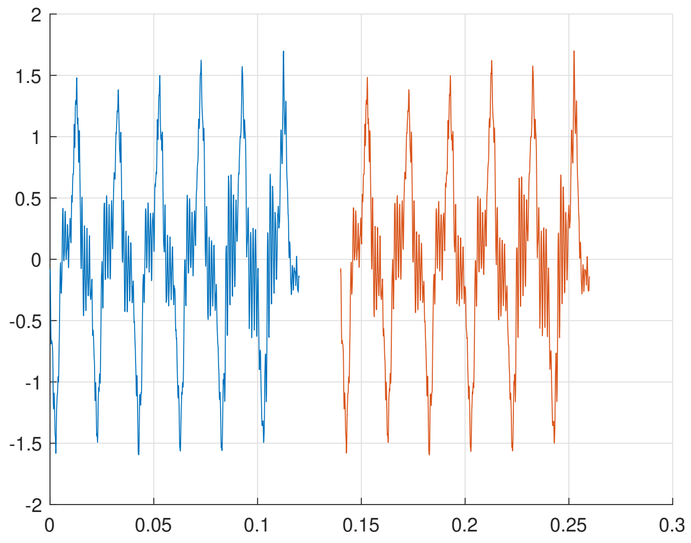


Figure 4.1: The first and second RSS resultant from splitting a measurement of stray magnetic field.

Table 4.1: Summary of datasets after splitting the OSS.

Scenario	Fault Type	Number of samples
1	ITSC	27750
	SE	14250
2	ITSC	9250
	SE	4750

Training, testing and cross-validation

The data was initially split into a training set and a testing set, with 67% of the data in the training set. The data were split into the aforementioned sets according to their OSSids, such that the RSSs obtained from the same OSS would not appear in several of the sets. After an initial data-analysis it was discovered that the measurements from the different sensor locations were not distinguishable and would cause target leakage in the machine learning. For measurements of ITSC, sensor location 1 and 3 was assigned the same OSSids and placed in the training set. Measurements from sensor position 2 was distinguishable and therefore used as testing data. For datasets containing SE faults, all samples obtained from the same machine state, regardless of sensor location, were given the same id. This was done since there was essentially no difference in the measurements originating from different sensor locations when subjected to static eccentricity.

The results were different procedures for splitting of the data depending on which fault was analysed. For detection of ITSC, all measurements from sensor location 1 and 3 was placed in the training data, while sensor location 2 was chosen as test data. For detection of SE, the OSSids were changed to reflect the discoveries regarding sensor location. The datasets were then split into a 70%-30% random split based on the OSSids. The test set functioned as a hold-out-set, and was only used to verify the performance of the algorithms.

Multi-Sensor Fusion

Multi-sensor fusion was performed by evaluating the different sensors as separate features. Sensors in this section refers to the type of sensor, not to be confused with sensor location. Features were calculated for the signals from each sensor then stored in a feature-dataframe with an ID representing the sample the features were extracted from. Different sensors representing the same machine state were then matched such that every sample used in the machine learning algorithm contained features from all sensors. To avoid data leakage due to sensor locations, the signals were matched based on sensor location. This means that the features obtained from the stray field sensor on location 1 were matched with features from the vibration sensor at location 1. Thus every sample in the dataset represented sensor fusion of features extracted from several different sensors.

The described methodology was preferred since the feature extraction methods had varying performance based on the signals. Calculating features from each signal independently then performing sensor-fusion of the features enabled feature extraction and selection based on the nature of the signal, i.e. stray and vibration did not need to contain the same features. This ensured that the optimal signal processing tools could be utilized on the relevant signal. Signal level fusion, such as non-linearly averaging the different measurements or evidence theory based fusion [13, 15], was not pursued in this thesis, as the above mentioned method was much simpler to implement and was thought to yield sufficient results. The proposed sensor fusion method was developed based on the waterfall fusion model and can be viewed as a black box, with data from several sources as input and single feature vector as output [12]. The sensor fusion method is illustrated in fig. 4.2, where the sensor fusion occurs in the feature extraction process.

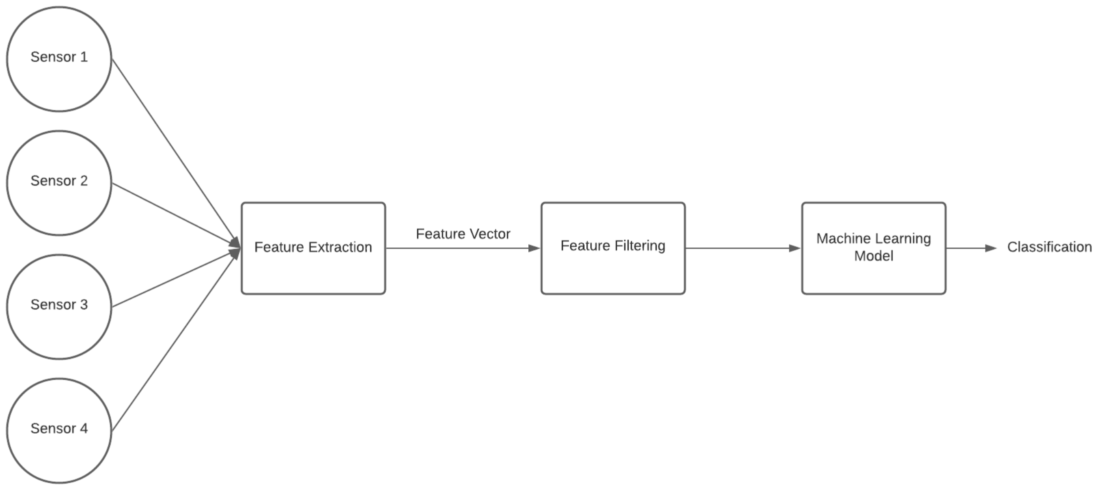


Figure 4.2: Block diagram visualizing the sensor-fusion process.

4.2 Feature Extraction

4.2.1 FFT

The frequency spectrum of the measurements was extracted through the fast Fourier transform using the Hanning windowing function. The Hanning window has been used frequently in the literature on similar signals, and proved an efficient and accurate method in work preceding this thesis [8, 9, 10]. In the literature, specific harmonics of the magnetic fields and vibration have been used as indices of various faults in electrical machines. These frequency harmonics appear at multiples of the rotational frequency of the machine, expressed in eq. (4.1) where f_{harm} is the frequency harmonic, p is the number of poles of the machine, k is an integer and f_e is the fundamental electrical frequency. These harmonics has shown a strong correlation to ITSC. As it was uncertain how the machine learning would respond to the different features and since there were multiple identifying harmonics in each of the signals, all harmonics following eq. (4.1) was included as features for the machine learning. The frequency spectrum was limited to 500Hz, as the low frequency content has been found to be most influential in fault detection. These features was later filtered based on correlation to the target value, which eliminated uninformative features.

$$f_{harm} = \frac{p+k}{p} f_e \quad , \quad k \in \mathbb{Z} \tag{4.1}$$

The usual method of detecting SE are based on a comparison of harmonic amplitudes from two sensors located 180 degrees apart in the radial axis on the generator. This was not possible based on the dataset and algorithm developed, however, SE are also found to influence the harmonics of a single sensor when compared to the healthy case. As such, harmonics of the rotational frequency was extracted for both ITSC and SE.

For the stray field measurements, the frequency spectrum was normalized by the maximum amplitude harmonic. This facilitate fault detection across different loads, as the amplitude variations due to loading are effectively filtered out by the normalization procedure. For the vibration signal it was found in a separate analysis that normalizing by power masks the fault indices. The vibration signal was instead normalized by the length of the signal, which was found to yield more consistent results in terms of fault detection. Aforementioned results came to light during analysis of different vibration sensors and post-processing of the measurements. This analysis has unfortunately not been included in the thesis. FFT of the stator voltage and current was performed in the same manner as the stray magnetic field.

4.2.2 Wavelet transform

DWT was performed using the pywavelt package in python. The Haar wavelet was used as mother wavelet with a decomposition of 10 levels. The DWT was implemented on each of the RSSs and the wavelet energies discussed in section 2.7 was extracted. To diminish end-effects in the DWT, each RSS was concatenated to 4 times its original length. All measurements were taken with a sampling frequency of 10kHz, resulting in the frequency bins seen in table 4.2.

4.2.3 TSFRESH

Time Series Feature Extraction Based on Scalable Hypothesis Tests was utilized to extract a large number of features from the time-series. The FFT features of TSFRESH was excluded as the most relevant frequency harmonics were extracted in the FFT analysis. FFT was performed outside of TSFRESH due to the inability to select frequencies of interest and tune the FFT algorithm to the specific signal at hand. The features extracted through TSFRESH are detailed in table C.1 in appendix C.1 [48].

Table 4.2: DWT frequency bands for a sampling frequency of 10kHz.

Levels	Frequency [Hz]
D1	5000-2500
D2	2500-1250
D3	1250-625
D4	625-312
D5	312-156
D6	156-78
D7	78-39
D8	39-19
D9	19-9
D10	9-4
A10	4-0

4.3 Exploratory Data Analysis (EDA)

The exploratory data analysis investigate the characteristics of the features used in machine learning. It is an evaluation of certain parameters such as the mean, standard deviation and correlation of the features to set the path for further feature engineering. The analysis was performed on the datasets of scenario 2, since this scenario contained all features used for both scenarios. In addition, a separate analysis was performed for each fault type, i.e. ITSC and SE. The type of fault significantly affects the features and their relevance in predicting the fault. As such, it was essential to analyse each fault separately. Principal component analysis was not included in this thesis due to its poor performance in [8].

4.3.1 ITSC

Mean and Standard Deviation

The mean and standard deviation of all features were calculated and are shown in fig. 4.3 and fig. 4.4, respectively. The FFT and DWT features all had approximately zero mean and standard deviation. The TSFRESH features varied greatly, both having significantly higher mean values as well as high standard deviation for certain features. Mean and standard deviation are important measures for some of the classifiers used this thesis. Classifiers like logistic regression and support vector machine require low variability in the mean and standard deviation of the features [26]. Since several of the features varied greatly from the norm in terms of both mean and standard deviation, standardisation should be performed on the dataset.

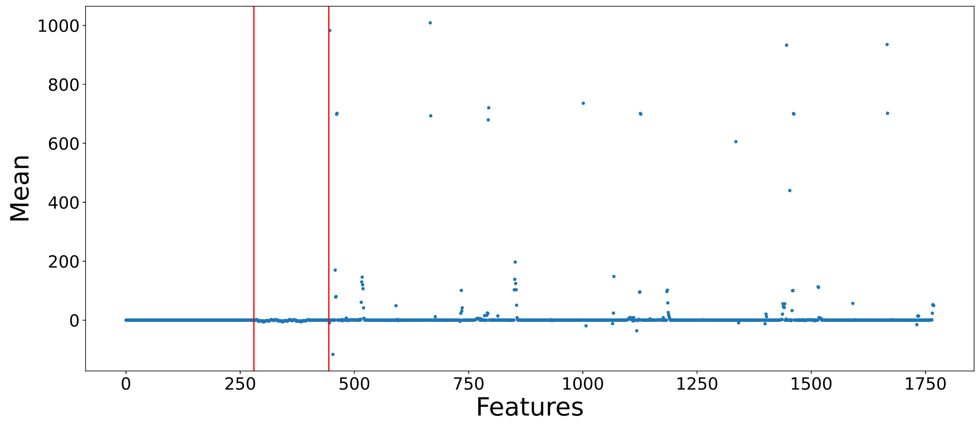


Figure 4.3: Mean of all individual features in the dataset for ITSC. The red lines separate the sections representing features from FFT, DWT and TSFRESH, from left to right. Inside each of these sections the features originating from stray magnetic field, vibration, current and voltage are in that respective order from left to right.

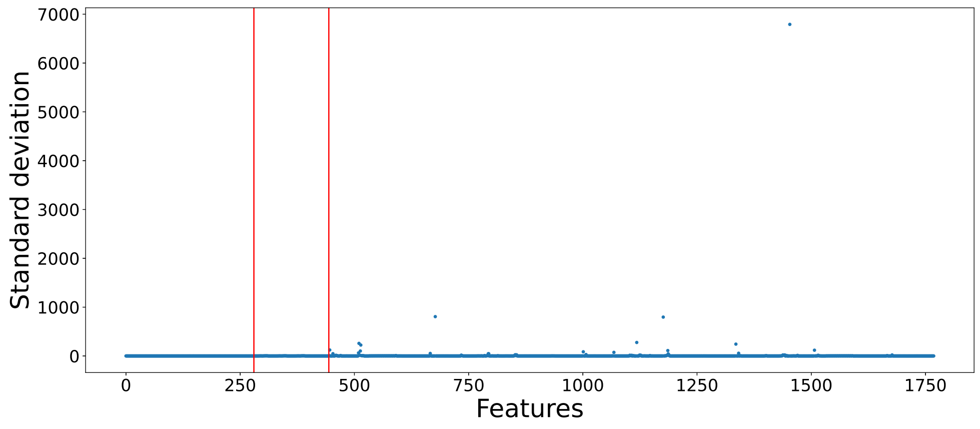


Figure 4.4: Standard deviation of all individual features in the dataset for ITSC. The red lines separate the sections representing features from FFT, DWT and TSFRESH, from left to right. Inside each of these sections the features originating from stray magnetic field, vibration, current and voltage are in that respective order from left to right.

Correlation

The Pearson correlation of each individual feature to the target values was calculated and are shown in fig. 4.5. The Pearson correlation describes the linear relationship between the individual features and the target labels, i.e. the correlation to faulty state of the machine. The Pearson correlation is a normalized covariance measure, thus it always falls between -1 and 1. There are no explicit rules of determining a significant correlation, as this depends on the problem at hand. In this thesis, an absolute correlation of above 0.3 were regarded as significant (inspired from [8]).

All feature extraction methods resulted in features of significant correlation, but also features of low correlation. The highest correlated features were extracted using TSFRESH. A large portion of the TSFRESH features were also largely uncorrelated. The FFT features were largely uncorrelated with a few exceptions. The DWT features produced many high correlated features. Interestingly, a large portion of these features were negatively correlated to the target label. This is a contradiction to what was found in [8], were the DWT energies were found to be strongly positively correlated to the number of short-circuited turns in the machine.

The large amount of insignificantly correlated features suggests that feature filtering is warranted. However, Pearson correlation shows only linear relationships, thus there might exist non-linear relationships in the features of low correlation. The 20 highest correlated features for ITSC are displayed in appendix B.2.

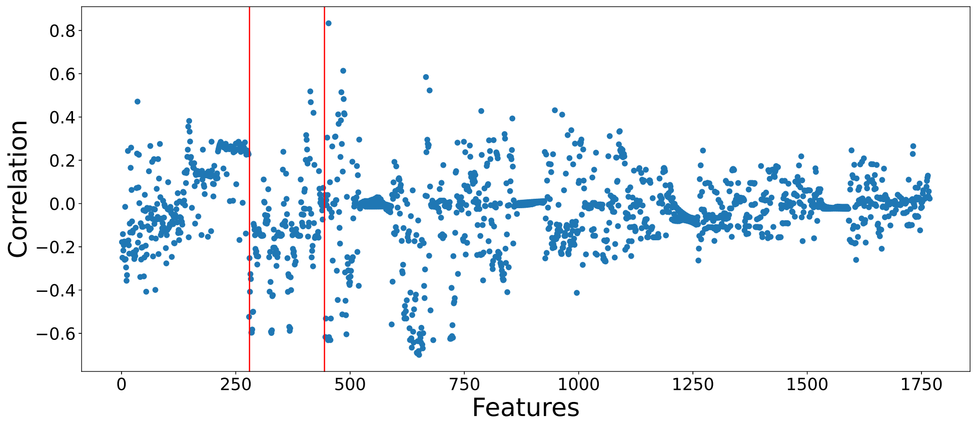


Figure 4.5: Correlation of individual features to the target label in the dataset for ITSC. The red lines separate the section representing features from FFT, DWT and TSFRESH, from left to right. Inside each of these section the features originating from stray magnetic field, vibration, current and voltage are in the respective order from left to right.

Confusion matrix

The confusion matrix show inter-feature correlation, i.e. the correlation between the individual features. Many correlated features warrant feature selection, since the system can most likely be described with fewer features. The correlation matrix are shown in fig. 4.6, where darker colours represents strong correlations.

The FFT features shows a relatively low correlation to other features. The FFT-features of the stator voltage shows a strong correlation among themselves and low correlation to other features. This can be seen in the darker rectangle at the intersection of the first red lines starting from the top left corner. The DWT features shows strong correlation to both other DWT features, the FFT features as well as some of the TSFRESH features. This might imply that fault detection can be performed using only the DWT features, since they show both a strong correlation to other features as well as to the target label. The TSFRESH features shows a varying inter-feature correlation. A group of the features resulting from TSFRESH have a strong correlation amongst themselves, but zero correlation to other features. This can be seen in the clear lines of white colour in the correlation matrix. Since many features have a strong correlation between themselves and a low correlation to the target label, feature selection procedures should be investigated. Feature selection were warranted regardless of the EDA due to the sheer size of the feature space, however, the EDA provides further justification for features selection.

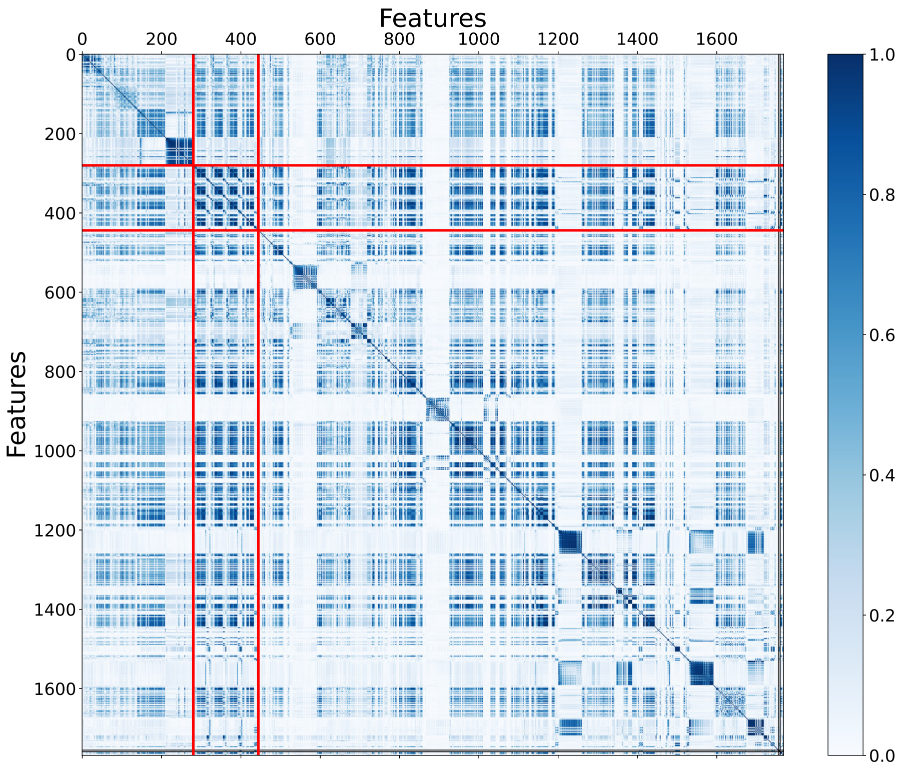


Figure 4.6: Correlation matrix for the dataset containing measurements of ITSC. The red lines represents the transition from FFT-, DWT- and TSFRESH-features from left to right.

4.3.2 SE

An equivalent analysis was performed on the dataset for SE fault and are summarized in this section. The plots of the mean, standard deviation, correlation and the correlation matrix can be found in appendix B.1.

As with the ITSC-EDA, there were outliers with a large variation in both mean and standard deviation in the SE-features as well. Especially the TSFRESH-features had large variations in the mean. This is resonable since both FFT and DWT had been standardised in the feature extraction procedure. Nonetheless, this result implies that scaling of the features are warranted.

The correlation plot shows a large number of highly correlated features resulting from all feature extraction methods. The FFT features show a larger correlation to SE than ITSC, and there are a significantly larger number of correlated features. The DWT-features shows largely the same pattern as for ITSC, with a large number of negatively correlated features. The highest correlated features results from TSFRESH, with some reaching about 0.8 and

0.9 in the Pearson correlation. This is very high correlation, suggesting that there is a linear relationship between the features and static eccentricity. The 20 highest correlated features for SE are displayed in appendix B.2.

The correlation matrix, showing the inter-feature correlation, shows relatively light colours indicating that a large number of the features are not highly correlated to each other. It also shows roughly the same pattern as for ITSC. This indicates that some of the features are correlated to each other regardless of fault, most likely due to the nature of the features. Since there are non-independent features in the dataset, feature filtering are warranted.

4.4 Feature Filtering

As identified in the EDA, several features were highly correlated while others might be uninformative. Reduction of the dimensions of the feature space were also considered beneficial due to the limited data. As the feature space increases in dimensions, the size of the dataset needed to train the machine learning algorithms to a sufficient degree increases. This phenomenon are referred to as the *curse of dimensionality*, and for some classifiers, like KNN, the required data increases exponentially with the dimensions of the feature space. In addition, the time needed to train and test the algorithms increases with the size of the feature space. Feature filtering is therefore justified and a paramount procedure for developing a functioning classifier based on the dataset at hand. Two methods of feature filtering have been applied, namely random forest and TSFRESH. Both methods were applied to the training data to avoid data leakage into the test set. This does however, increase the performance of the classifiers on the training data, since feature selection was based on this data.

Random Forest

A random forest classifier with 1000 estimators and the Gini index as splitting criterion was trained on the training data. The feature importance was extracted and used to filter out the features deemed by the classifier as less-important. All features of less than mean importance, based on the Gini impurity index, was excluded from the dataset. The results can be seen in table 4.3.

TSFRESH

TSFRESH extracts the feature relevance through the scalable hypothesis test. This method evaluates each feature independently with respect to its significance for predicting the target, which results in a vector of p-values denoting the significance of each feature [48]. The p-values are then evaluated based on the false discovery rate through the Benjamini-Yekutieli procedure. The false discovery rate (FDR) in hypothesis testing is the expected proportion of erroneous rejections among all rejections [49]. A feature is deemed relevant if the null-hypothesis is rejected based on the false discovery rate. The FDR level was set to the default of 0.05. This represents the theoretical expected percentage of irrelevant features in the dataset. The results from filtering using the TSFRESH-algorithm can be seen in table 4.3.

Summary

Table 4.3 displays a summary of the feature selection process. Since the features for ITSC and SE differ significantly, it was decided to filter each dataset separately. In addition, the sensor fusion scenarios mentioned in previous chapters was evaluated independently. This resulted in a total of 8 datasets. Datasets of subscript A was filtered through Random Forest, while datasets of subscript B was filtered by TSFRESH.

Random forest resulted in a significantly lower number of relevant features and eliminated around 80% of all features. Depending on the performance of the different datasets, this could imply that many of the features calculated are irrelevant for detection of faults in synchronous generators. TSFRESH eliminated about 15% of the features. This was most likely due to the settings used in the algorithm. If a higher value for the FDR were used, more features are likely to have been eliminated. A low number of features are preferred if the performance is constant.

Table 4.3: Summary of datasets after feature selection.

Scenario	Dataset	Filtering method	Fault type	# Features before filtering	# Features after filtering
1	A1	Random Forest	ITSC	886	181
	A2	Random Forest	SE	889	168
	B1	TSFRESH	ITSC	886	756
	B2	TSFRESH	SE	889	728
2	A3	Random Forest	ITSC	1769	273
	A4	Random Forest	SE	1769	172
	B3	TSFRESH	ITSC	1769	1396
	B4	TSFRESH	SE	1769	1322

4.5 Algorithm Selection and Data Set Performance

This section describes the initial results from testing multiple machine learning models. To select the best algorithm for the classification task, several classifiers was trained and tested through a 5-fold CV split of the training data. The 5-fold-CV split was chosen due to the relatively low number of samples and to limit sampling bias. The 5-fold-CV split was implemented to split based on OSSids to avoid target leakage. The procedure was repeated for each of the datasets mentioned in table 4.3 to evaluate the best dataset for each task. Following is a list of the classifiers tested in this section.

- Logistic Regression
- KNN
- SVM (rbf)
- SVM (Linear)
- XGBoost
- Neural Network
- Stacking classifier consisting of: Logistic Regression, SVM (Linear), XGBoost and Neural Network with logistic regression as meta-classifier.

The stacking classifier was constructed based on the highest performing classifiers in [8], and consisted of logistic regression, SVM using the linear kernel, XGBoost and an MLP classifier (Neural Network).

The hyperparameters used in this section was based on "rule of thumb" estimates for each classifier, and were adapted from [8]. This means the results in this section are expected to be limited due to the non-optimized hyperparameters. The non-default hyperparameters used in this section are presented in table table 4.4 (inspired from [8]).

Table 4.4: Initial hyperparameters for the classifiers. Classifiers using default parameters have not been included in the list [8].

Classifier	Hyperparameters	Settings
KNN	K	20
	Weight	Uniform
SVM (rbf)	Kernel	Radial basis function
	Gamma	$\frac{1}{\text{Number of features}}$
XGBoost	Et	0.3
	Max depth	6
Neural Net	Hidden layers	2
	Neurons in 1st layer	200
	Neurons in 2nd layer	100
	Neurons in 3rd layer	14
	Activation function	ReLU

Standardisation

On the basis of the EDA, scaling/standardisation of the features was imperative. The features were scaled using the StandardScaler from the sklearn.preprocessing module. The StandardScaler functions similar to classifier, and was trained on the training data. The scaling was implemented within the 5-fold-CV split, such that in each split the features were scaled based on the training data of that specific split. This eliminated target leakage due to standardisation.

Imbalance of classes

As previously discussed, the datasets were highly imbalanced with the majority of samples representing faulty machine. This was addressed by implementing a combination of under-sampling of the majority class and over-sampling of the minority class. It was desired to pursue over-sampling methods due to the limited data. Under-sampling would reduce the dataset, thus potentially depreciating the training of the machine learning algorithms. The combination of over- and under-sampling limits the weaknesses of each method.

The dataset was first over-sampled using the SMOTE algorithm, then under-sampled through the Edited Nearest Neighbour (ENN) algorithm. By implementing the under-sampling after the SMOTE algorithm, much of the noisy samples generated by SMOTE were eliminated. The functionality of the ENN algorithm effectively cleans up the "neighbourhoods" in the dataset by eliminating ouliers and inliers. The remaining dataset therefore likely consisted of more defined decision boundaries, which aids in classification.

The SMOTE-ENN algorithm was implemented in a similar fashion as the StandardScaler. At each fold in the 5-fold-CV split, the training data was resampled to adjust the balance of classes. The number of samples added by the algorithm varied depending on the dataset and the CV-folds, and was generally in the range of 5000- 10000 samples. The minority class was the healthy case throughout all sets, thus the synthetic samples represented healthy samples. After resampling, the training data consisted of an equal percentage of all classes.

Evaluation Metrics

The balance of the training set was adjusted using the SMOTE-ENN algorithm, however, the test- and cv-sets remained unbalanced. As discussed, some metrics gives a false impression of good performance in the presence of unbalanced classes. In the mentioned dataset, the minority class were represented by the Boolean false value, i.e. negative class. Based on this, ROC AUC and specificity were the chosen metrics for evaluating the performance of the algorithms. These metrics are bound to the negative class, thus producing a more accurate picture of the performance in the face of a scarce sample of negative classes. Accuracy, precision and recall are not as relevant for this dataset since they measure the prediction performance of the positive class. Since the positive class represents a significant portion of the test set, any classifier that performs well on the positive class would yield a high accuracy, precision and recall. This is valid for the F1-score as well, since it is a product of recall and precision.

Results

The average performance of the classifiers across the 5-fold CV split with "rule-of-thumb" hyperparameters, scaled features and balanced dataset are presented in table 4.5 for scenario 1 and table 4.6 for scenario 2.

Table 4.5: Results from testing multiple classifers using sensor-fusion scenario 1.

Data set	Classifier	Accuracy	Specificity	F1-score	ROC AUC
A1	Logistic Regression	0.9392	0.9264	0.9592	0.9430
	KNN	0.8065	0.7360	0.8714	0.7827
	SVM (rbf)	0.9374	0.9220	0.9577	0.9398
	SVM (linear)	0.9401	0.9267	0.9603	0.9441
	XGBoost	0.8973	0.8568	0.9332	0.8966
	Neural net	0.9200	0.8475	0.9476	0.9010
	Stack	0.9380	0.8495	0.9593	0.9141
	Average classifier score	**0.9112**	**0.8664**	**0.9413**	**0.9030**
B1	Logistic Regression	0.9226	0.8400	0.9501	0.9069
	KNN	0.4606	0.4425	0.5778	0.4607
	SVM (rbf)	0.8777	0.7015	0.9239	0.8284
	SVM (linear)	0.9365	0.8760	0.9581	0.9267
	XGBoost	0.8886	0.8707	0.9265	0.8951
	Neural net	0.8429	0.6981	0.8995	0.8081
	Stack	0.8925	0.6659	0.9348	0.8246
	Average classifier score	**0.8316**	**0.7278**	**0.8815**	**0.8072**
A2	Logistic Regression	1	1	1	1
	KNN	1	1	1	1
	SVM (rbf)	0.9530	1	0.9677	0.9687
	SVM (linear)	1	1	1	1
	XGBoost	0.9983	1	0.9989	0.9989
	Neural net	1	1	1	1
	Stack	1	1	1	1
	Average classifier score	**0.9930**	**1**	**0.9952**	**0.9954**
B2	Logistic Regression	1	1	1	1
	KNN	0.9997	1	0.9998	0.9998
	SVM (rbf)	0.9987	1	0.9991	0.9991
	SVM (linear)	1	1	1	1
	XGBoost	0.9983	1	0.9989	0.9989
	Neural net	1	1	1	1
	Stack	1	1	1	1
	Average classifier score	**0.9995**	**1**	**0.9997**	**0.9997**

Table 4.6: Results from testing multiple classifers using sensor-fusion scenario 2.

Data set	Classifier	Accuracy	Specificity	F1-score	ROC AUC
A3	Logistic Regression	0.5737	0	0.7291	0.3347
	KNN	0.5789	0	0.7333	0.3377
	SVM (rbf)	0.5994	0	0.7496	0.3497
	SVM (linear)	0.5749	0	0.7300	0.3353
	XGBoost	0.5869	0	0.7396	0.3423
	Neural net	0.5829	0	0.7365	0.3400
	Stack	0.6434	0	0.7830	0.3753
	Average classifier score	**0.5914**	**0**	**0.7430**	**0.3450**
B3	Logistic Regression	0.6189	0	0.7646	0.3610
	KNN	0.5457	0	0.7061	0.3183
	SVM (rbf)	0.7229	0	0.8391	0.4217
	SVM (linear)	0.6057	0	0.7544	0.3533
	XGBoost	0.6057	0	0.7544	0.3533
	Neural net	0.7114	0	0.8314	0.4150
	Stack	0.6909	0	0.8172	0.4030
	Average classifier score	**0.6430**	**0**	**0.7810**	**0.3751**
A4	Logistic Regression	1	1	1	1
	KNN	1	1	1	1
	SVM (rbf)	1	1	1	1
	SVM (linear)	1	1	1	1
	XGBoost	1	1	1	1
	Neural net	1	1	1	1
	Stack	1	1	1	1
	Average classifier score	**1**	**1**	**1**	**1**
B4	Logistic Regression	1	1	1	1
	KNN	1	1	1	1
	SVM (rbf)	0.7770	1	0.8254	0.8513
	SVM (linear)	1	1	1	1
	XGBoost	1	1	1	1
	Neural net	1	1	1	1
	Stack	1	1	1	1
	Average classifier score	**0.9681**	**1**	**0.9751**	**0.9788**

4.5.1 Scenario 1

Dataset A1 & B1 - ITSC

The datasets resulting from feature selection using random forest consistently outperformed the datasets using TSFRESH feature selection for all classifiers and on all evaluation metrics. The average performance of all classifiers were significantly higher for the RF-datasets, with an average increase in ROC AUC of about 10%. In addition, the RF-datasets contained far less features. Both in terms of performance and size, the RF-datasets reigns supreme. This result implies that Random Forest is the preferred method of feature selection.

All classifiers scored high in accuracy, presicion, recall and F1-score. As discussed previously, any classifier that has a decent ability to detect the positive class will score high in these metrics in the face of an imbalanced dataset. More interestingly, the classifiers had a high score on all metrics across the CV-folds. As previously discussed, the training data consisted of a large number of identical samples which makes it likely the algorithm will overfit. If the algorithm overfits in the CV-folds, it is likely to have a high accuracy since there is little diversity in the data. In addition, there was some data leakage in the CV-folds since feature selection had been performed prior to the CV-fold split. The features used were selected based on their relevance for classification tasks on the entire training set. Since the training set was split into CV-folds, each subset contained some information about the entire set. It is therefore expected a significantly lower performance on the test set both due to data leakage as well as the high possibility of overfitting.

The highest performing classifiers were logistic regression, SVM using the linear kernel, XGBoost and the stacking classifier. KNN gave the worst performance on all metrics, with a performance significantly lower than the other classifiers. Among the metrics, specificity was consistently the lowest followed by ROC AUC. This confirms that these metrics are the most suited for imbalanced classes. The low score in specificity means the classifiers have a hard time detecting the healthy case, since there are few true negatives while proportionally higher rates of false positives. The ROC AUC score signals the same conclusion, that the classifiers have a hard time separating the classes. The classes was balanced before training the algorithms. Since the results are still poor after balancing, it signals that the balancing methodology did not work as intended. Even after balancing the classes, the algorithms are not given enough input to properly distinguish healthy from faulty.

Dataset A2 & B2 - Static Eccentricity

The low amount of measurements and large amount of identical samples proved to be a challenge when performing the k-fold-CV split. Certain splits would only contain one of the classes, making it impossible to evaluate those splits. It was therefore decided to implement a new splitting algorithm for the datasets containing measurements of eccentricity (dataset A2 and B2). This was done both to ensure representative splits as well as to ensure there was no target leakage in the splits. Instead of using k-fold-cv split, a simple train-, test-, CV-split was used with the datasets reflecting 60%, 20% and 20% of all available data, respectively.

Initial testing with this split revealed a perfect score among all classifiers. This was a sign of data leakage, thus a procedure to identify the leak followed. The procedure was to split the data manually into a train-, test- and CV-set in iterations, where each iteration included a different split. The procedure are described as follows:

1. Split the data into a training set, a testing set and a CV-set.
2. Select features through RF, since this proved to be the optimal feature selection method.
3. Balance the training set based on SMOTE-ENN.
4. Train the classifiers using the balanced training set and evaluate the performance on both the CV-set and the test set.
5. If there was data leakage, repeat from 1 with a new split.

At each iteration the measurements reflected in the different sets was recorded to analyse what could be the source of the target leakage. The datasets were always split according to their OSSid, and the splits were designed such that measurements from different sensor locations with the same machine state were all placed in the same split, i.e. either in the train-, test- or the CV-set. It was therefore impossible that the score was a result of target leakage based on different sensor locations.

Through the iterations, different combinations of loads were placed in the respective sets. All though not all combinations were tested, enough iterations were run to give confidence in that the load was not a source of data leakage. In addition, there was no data leakage due to the scaling of features or feature selection since this was implemented on the the training set only. Since the possible factors of leakage had been exhausted the thesis concluded that this result does not reflect data leakage.

The other possibility of a perfect score is that there exist a clear relationship between the samples and the classes. Say for example there is a linear relationship between eccentricity and the amplitude of the measured signal. If this is the case, it is reasonable to expect a perfect score on the classifiers, since all of them are capable of modelling linear relationships. Since the data leakage analysis yielded no results, the only conclusion that could be drawn from this is that the performance of the classifiers reflect the true ability to detect static eccentricity.

4.5.2 Scenario 2

The same procedure as for scenario 1 was applied for scenario 2, however, the low amount of data for this scenario became an apparent problem. The results from the classifiers were prone to an extremely large sampling bias, and the performance of the classifiers ranged from 40% accuracy to 80% accuracy depending on the split. Due to the few measurements taken, the training- and CV-sets could never be representative of every machine state. Thus the results depended heavily upon the split, and none of the splits had sufficient data for the algorithms to obtain a good fit.

Due to the low amount of data, the CV-set and test-set effectively consisted of a maximum of 6 measurements. This implies that there was usually a single measurement of the healthy case in these sets, and at most two. This means that the performance of the classifiers were not fully representative, since only a limited number of machine states were tested in each split.

Based on these conditions it was decided to drop all efforts of developing a machine learning algorithm based on scenario 2. The low amount of data made the task impossible, and the possible ways of extracting more data was exhausted. Since scenario 2 was decided to be eliminated from further analysis, all discussion in the remaining thesis refers to scenario 1 unless stated otherwise.

4.5.3 Conclusion of Classifier and dataset selection

For detection of ITSC, datasets filtered through Random Forest significantly outperformed those filtered through TSFRESH, both in terms of classifier performance and number of features. Based on these results, all remaining work was conducted on datasets filtered through random forest. All classifiers performed well on this dataset, most likely due to the low diversity within the training set. This condition is also likely to cause overfitting.

For detection of SE, all classifiers had a near perfect score across a 5-fold CV split. The split was performed in a fashion that eliminated target leakage due to OSSids and sensor location. Target leakage could have occured due to loading, however, this was seen as highly unlikely. Further developing of machine learning algorithms for detection of static eccentricity was therefore not warranted since the performance of the algorithms was peaked.

Scenario 2 was eliminated from further analysis due to low amount of data. This means that the only dataset that was used in the following sections were $A1$, representing detection of ITSC based on scenario 1 filtered through Random Forest.

4.6 Hyperparameter Selection

The results from the previous section was from mostly default hyperparameters. Out-of-the-box classifiers will most likely give limited results since the hyperparameters are not fitted to the dataset and task at hand. As such, this section seeks to soft-optimize the hyperparameters through a grid-search.

Hyperparameters of machine learning algorithms refers to variables determining the architecture and general functionality of the algorithm. The number of combinations of hyperparameters within the classifiers are essentially infinite, thus testing all combinations are both impossible and unreasonable. A grid-search was therefore performed to soft-optimize the parameters. The grid-search evaluates the performance of classifiers over a limited set of hyperparameter combinations.

Due to limited knowledge in machine learning, the input of the grid-search was adapted based on the results in the preceding thesis [8]. For certain of the classifiers, the optimal parameters were found at the boundaries of the grid search. It was therefore natural to extend the grid-search to evaluate if parameters outside the grid-search in [8] could improve the performance. A broader grid-search with fewer increments were chosen, both to decrease computation time and to investigate parameters beyond the scope in [8].

The classifiers were chosen based on the performance in section 4.5. KNN was excluded from this analysis due to the poor performance. XGBoost was also excluded due to an inordinate many hyperparameters and training time. To do a sufficient grid-search for XGBoost, the thesis limited the hyperparameter-combinations down to 1500. This would however take more than 350hours to complete which was not within the budget of this thesis. The large number of hyperparameter combinations are due to XGBoost being an ensemble classifier, which has important hyperparameters both for the base learners as for the meta-classifier.

The stacking classifier was excluded from the grid search for the same reason.

Since the likelihood of overfitting was a large concern considering the dataset, extra emphasis was put on parameters limiting overfitting, such as regularization parameters. Regularization parameters in algorithms are designed to limit overfitting. In addition, ROC AUC was used to evaluate the performance of different hyperparameter combinations. Not only are ROC AUC robust in the face of imbalanced classes, it has also been found to be resistant to overfitting. The ROC AUC are constructed from the true positive rate (TPR) and the false positive rate (FPR). To obtain a high ROC AUC one must maximize the TPR while minimizing FPR, thus the function works as a regulator. However, overfitting can still easily occur in a grid-search, thus the evaluation of the optimal parameters are a necessity. The hyperparameters and classifiers used in the grid search are expressed in table 4.7. The grid search was performed using a 5-fold CV split.

Table 4.7: Hyperparameter settings for optimization of classifiers through grid-search (inspired by [8]).

Classifier	Hyperparameter	Setting	Description
Logistic Regression	C	10^k for $k = -10, -9.5, ..., 10$	Inverse of regularization strength
	Penalty	L1, L2	The norm used in penalization of the cost function
Support Vector Machine	C	0.1, 1, 5, 10, 20, 100, 1000	Inverse of regularization strength
	gamma	'scale' 10, 2, 1, 0.5, 0.1, 0.01, 0.001	Kernel coefficient for the rbf and sigmoid kernel
	kernel	rbf, linear, sigmoid	The kernel function
Neural Network	hidden_layer_sizes	(25, 25, 3), (50, 25, 3), (200, 100, 14), (300, 150, 21)	Size and number of hidden layers
	activation	identity, logistic, tanh, ReLu	The activation function
	batch_size	300, 200, 100, 50	Size of minibatches
	max_iter	100, 200, 500, 1000, 2000	Maximum number of training iterations (epochs)

The optimal hyperparameters found through the grid-search using ROC AUC as evaluation metric are expressed in table 4.8. For logistic regression, the $l1$ penalty function has been removed from the algorithm, thus $l2$ was the only penalty actually tested in the grid-search. The regularization parameter $logreg\_C$ showed an extremely large value as the optimal. This parameter represents the inverse strength of the regularization term, meaning a high value gives little regularization. Without regularization the algorithm are likely to overfit. The grid-search therefore suggest that to obtain the highest performance in the CV-split, there should be given little emphasis on regularization, thus allowing the algorithms to overfit. This confirms the suspicion about overfitting. Since the training data has very low diversity, an overfitted algorithm would perform very well and thus be chosen as the optimal algorithm by the grid-search.

For the SVM classifier, the radial basis function was chosen as the optimal kernel function, even though the linear kernel significantly outperformed the rbf-kernel with default parameters. This shows that algorithms should not be chosen based on out-of-the-box performance, as the performance are likely to change after optimizing hyperparameters. As with the logistic regression classifier, the regularization terms was chosen as the highest value included in the grid search. This again signals that the grid-search optimization leans towards allowing overfitting. It is possible that a higher value would have given better performance since the svm_C was found at the border of the grid-search. This would however push the algorithm further in the direction of overfitting and would therefore be sub-optimal in the long run.

For the neural network the lowest number of epochs were chosen as optimal. This indicate that the neural network learns quickly and that including more iterations does not reduce the classification error. The highest performing activation function was the identity-function. This makes the MLP linear, indicating a linear relationship between the features and the target labels. The rest of the parameters are difficult to interpret since the neural network is a very complex algorithm. It could be stated that the architecture chosen was among the most complex included in the grid search, however, the architectures included in the grid search was quite limited. For a thorough grid-search, more parameters and architectures should have been included for the Neural network, however this was considered a to computational heavy task for this thesis. The goal of the grid-search was to soft-optimize the parameters, which has been accomplished to a sufficient degree.

Table 4.8: Resultant optimal hyperparameters found through a grid-search using ROC AUC as evaluation metric.

Classifier	**Hyperparameter**	**Value**
Logistic Regression	logreg_C	1000000000.000
Logistic Regression	logreg_penalty	l2
SVM	svm_C	1000
SVM	svm_gamma	0.001
SVM	svm_kernel	rbf
Artificial Neural Network	neural_net_activation	identity
Artificial Neural Network	neural_net_batch_size	50
Artificial Neural Network	hidden_layer_sizes	(200, 100, 14)
Artificial Neural Network	neural_net_max_iter	100

The optimal hyperparameters were implemented and the algorithms were tested on the 5-fold-CV split. The average results from the optimised algorithms tested on the CV-split are presented in table 4.9. The ROC AUC significantly increased as a result of the grid-search. In addition the accuracy along with the other evaluation metrics increased. This suggests that the predictive performance on all classes increased as a function of the grid search.

Table 4.9: Average results from classifiers tested using a 5-fold-CV split and hyperparameters optimized through grid-search.

	Accuracy	Recall	Precision	Specificity	F1-score	ROC AUC
Logistic Regression	0.9554	0.9628	0.9801	0.9639	0.9705	0.9634
SVM (rbf)	0.9329	0.9515	0.9645	0.9296	0.9558	0.9406
Neural net	0.9559	0.9638	0.9807	0.9644	0.9711	0.9641

4.7 Performance of algorithms on hold-out data

The results from the grid-search are overly optimistic due to both data leakage through feature selection and data leakage resulting from optimizing the classifiers on the same set it was tested on. To fully assess the performance of the algorithms they were tested on the test-set. This set had been kept outside all processes to avoid data leakage, and consisted of measurements from sensor location 2. To avoid confusion, this set will be referred to as the *hold-out set*. This dataset differed from the measurements included in the training set, which contained sensor location 1 and 3. These locations was influenced by the axial flux of the generator to a higher degree due to the sensor location. Based on these conditions it is expected a significantly lower performance on the hold-out set. The results from testing the algorithms on the hold-out set set are shown in table 4.10.

Table 4.10: Results from testing all classifiers on the hold-out set. Logistic regression, SVM (rbf) and Neural net were implemented using optimized hyperparameters. The stacking classifier consists of optimized logistic regression, SVM (rbf) and Neural net, with logistic regression as meta classifier

	Accuracy	Recall	Precision	Specificity	F1-score	ROC AUC
Logistic Regression	0.7376	0.7325	0.9288	0.7594	0.8191	0.7460
KNN	0.6846	0.8336	0.7893	0.0463	0.8108	0.4399
SVM (rbf)	0.7576	0.7715	0.9164	0.6983	0.8377	0.7349
SVM (linear)	0.7268	0.6972	0.9533	0.8537	0.8054	0.7755
XGBoost	0.8108	1.0000	0.8108	0.0000	0.8955	0.5000
Neural net	0.7582	0.7813	0.9075	0.6589	0.8397	0.7201
Stack	0.7963	0.8309	0.9100	0.6480	0.8687	0.7395

There was a significant decrease in performance when the algorithms were tested on the hold-out set, with all classifiers providing an underwhelming performance. One likely reason for this is overfitting. As discussed in many of the previous sections, overfitting were a large concern with this dataset. This is reflected in the results, as all classifiers had a high performance on across the CV-folds, but a subpar result on the test set. An accuracy or ROC AUC of about 80% means that 20% of the classes are misclassified, which is inadequate for industrial application. This could have been fixed by implementing regularisation parameters in the algorithms, however, efforts taken to optimize the performance of the classifiers for the test set would lead to overly optimistic estimates of the performance. As such, further efforts of improving the algorithms was not pursued.

The performance of the classifiers reflects the dataset. A poor dataset gives poor performances. The main problem of this dataset was the lack of diversity in the samples. The data was split to ensure no target leakage and to have a representative test set. This methodology caused the test set to be significantly different from the training set due to the sensor locations used when extracting the signals. In addition, most of the samples in the training set was indistinguishable. These are prime conditions for an overfitted algorithm with poor generalising capabilities.

It was of interest to investigate precisely which samples was misclassified. Information about the samples had been stored for verification purposes throughout the development of the model. This information was matched with the predictions of the highest performing classifier. The prediction accuracy for each case using SVM with the linear kernel are shown in table 4.11 .

Table 4.11: Prediction accuracy of SVM (linear) on the hold-out set for each fault type.

Machine State	Accuracy
Healthy	85.37%
1 Turn SC	64.20%
2 Turn SC	33.94%
3 Turn SC	67.00%
7 Turn SC	83.47%
10 Turn SC	100%

This result shows that the algorithm primarily struggles to distinguish between healthy case and the less severe cases of ITSC, while the severe faults are detected at a high accuracy. A large portion of the low severity faults go undetected. This suggest that by eliminating low severity cases from the dataset, one could possibly achieve a very good performance.

4.8 Improving performance

The dataset was split into training and testing set based on sensor location. This was found to be a sub-optimal solution, since it caused overfitting and a poor test result. The training set consisted of sensor location 1 and 3, which had a larger influence of the axial flux. These measurements differed from the test set which contained sensor location 2. Sensor location 2 was much less affected by the axial flux. As such, the performance of the algorithms suffered both due to overfitting and due to the test set being too different from the training set. A new split was tested in an attempt to remedy these problems. In this split, sensor location 1 and 3 was given the same OSSid due to their similarities. The data set was then split randomly based on the OSSids. This ensured that both the training and the test set contained measurements from all sensors. However, not all cases was represented in both sets. This means that for a specific load and fault state, that specific case would only occur in one of the sets. This resulted in some of the folds in CV-split not containing samples of the healthy case, thus ROC AUC and specificity was not defined for the CV-folds.

The resultant datasets was tested in a similar fashion as in the previous section. First a 5-fold-CV split was done on the training set, and the average performance was evaluated. Then the algorithms was trained on the entire training set and tested on the test set. The results are shown in table 4.12

Table 4.12: Results from testing the classifiers from section 4.5 with the new splitting strategy.

Results from 5-fold CV split						
Classifier	Accuracy	Recall	Precision	Specificity	F1-score	ROC AUC
Logistic Regression	0.8138	0.8908	0.8964	ill-defined	0.8835	ill-defined
KNN	0.7052	0.7784	0.8588	ill-defined	0.8085	ill-defined
SVM (rbf)	0.7946	0.9013	0.8526	ill-defined	0.8700	ill-defined
SVM (linear)	0.8238	0.9069	0.8975	ill-defined	0.8899	ill-defined
XGBoost	0.8361	0.9323	0.8798	ill-defined	0.8980	ill-defined
Neural net	0.8006	0.9077	0.8599	ill-defined	0.8761	ill-defined
Stack	0.8553	0.9819	0.8685	ill-defined	0.9134	ill-defined
Average classifier score	**0.8042**	**0.8999**	**0.8734**		**0.8771**	
Results from hold-out-set						
Logistic Regression	0.9492	0.9510	0.9807	0.9440	0.9656	0.9475
KNN	0.9480	0.9898	0.9436	0.8227	0.9662	0.9062
SVM (rbf)	0.9579	0.9880	0.9572	0.8676	0.9724	0.9278
SVM (linear)	0.9457	0.9560	0.9711	0.9147	0.9635	0.9353
XGBoost	0.9546	0.9963	0.9460	0.8293	0.9705	0.9128
Neural net	0.9610	0.9879	0.9612	0.8804	0.9744	0.9341
Stack	0.9529	0.9999	0.9410	0.8120	0.9695	0.9059
Average classifier score	**0.9527**	**0.9812**	**0.9573**	**0.8672**	**0.9689**	**0.9242**

As expected, the new split resulted in a decrease in the performance across the CV-folds. This is due to the training set having more diversity. In addition, not all the classes were represented within the folds. This leads to a decrease in performance since the algorithms are not trained on the classes they are predicting. The decrease in performance across the CV-folds was not a major issue, since it could be improved through optimizing the classifiers. The split could also be optimized to be more representative of the classes within the dataset. This was not pursued since the main goal of this section was to show how a different splitting methodology can yield a more generalizable algorithm, thus the important results in this section are those obtained on the hold-out-set.

The test results from the hold-out-set show an increase in both accuracy, specificity and ROC AUC. This indicates an improved classification performance, with both a higher TPR and a lower FPR. The specificity and ROC AUC are the preferred metrics for this dataset. Having both these metrics reach about 95% provides confidence that the classifier can distinguish the classes at a high degree. The results shows that the classifiers have good capabilities of detecting 1,2,3,7 and 10-turn short-circuit faults regardless of load conditions.

The results confirms the weaknesses of the previously implemented splitting strategy, and shows that by use of sensor-fusion and a proper splitting strategy it is possible to achieve a high performing classifier for fault detection of ITSC. For fault detection of ITSC, the proposed classifier are Logistic regression, using the splitting methodology of this section. This classifier was among the highest scoring on all metrics. In addition it scored the highest in both specificity and ROC AUC which are the preferred metrics. If a high detection rate are desired, at the cost of some false positives, the multi-layer perceptron classifier is recommended. This had a close to 100% recall, meaning nearly all faulty cases were detected.

Table 4.13 shows the improvement in the performance of logistic regression by including all sensor locations in both the training set and the test set. The classifier has a high accuracy and the splitting methodology used in this section are the preferred methodology.

Table 4.13: Comparison of splitting strategies using the Logistic regression regression classifier.

Splitting strategy	Accuracy	Recall	Precision	Specificity	F1-score	ROC AUC
Splitting by sensor location	0.7376	0.7325	0.9288	0.7594	0.8191	0.7460
Stochastic split based on OSSids	0.9492	0.9510	0.9807	0.9440	0.9656	0.9475

4.9 Influence of load

To evaluate the influence of load the no-load measurements was omitted from the dataset. The logistic regression classifier was trained and tested on this dataset, and the results are displayed in table 4.14. As seen in the results, omitting the no-load measurements increased the performance with approximately 3%. This shows that the loading of the generator does affect the performance of the algorithms. Without no-load measurements, the algorithm had zero false positives.

Table 4.14: Performance of logistic regression after omitting the no-load measurements from the dataset

Evaluation metric	Performance
Accuracy	0,9833
Recall	0,9792
Precision	1
Specificity	1
F1-score	0,9895
ROC AUC	0,9896

4.9.1 Feature Importance

The logistic regression classifier and XGBoost provides the ability of extracting feature importance. The logistic regression classifier determines weights for each of the features during training, which correspond to the importance of that feature for classification. For XGBoost, the variable importance measure described in section 2.4.4 were extracted. The most important features for logistic regression and XGBoost are displayed in table 4.15 and table 4.16, respectively.

Table 4.15: The 20 most relevant features for the Logistic Regression classifier from section 4.8. Stray and Vibration in the feature column refers to features originating from the stray field sensors and vibration sensors, respectively.

Rank	Feature	Description
1	Stray_FFT_300.1 Hz	FFT, frequency harmonic of 300.1 Hz
2	Stray_FFT_307.2 Hz	FFT, frequency harmonic of 307.2 Hz
3	Stray_FFT_107.2 Hz	FFT, frequency harmonic of 107.2 Hz
4	Vibration_FFT_114.3 Hz	FFT, frequency harmonic of 114.3 Hz
5	Stray_FFT_350.1 Hz	FFT, frequency harmonic of 350.1 Hz
6	Vibration_FFT_300.1 Hz	FFT, frequency harmonic of 300.1 Hz
7	Stray_FFT_292.9 Hz	FFT, frequency harmonic of 292.9 Hz
8	Vibration__c3__lag_1	The c3 statistics to measure non linearity in the time series
9	Stray_FFT_92.9 Hz	FFT, frequency harmonic of 92.9 Hz
10	Stray_FFT_42.9 Hz	FFT, frequency harmonic of 42.9 Hz
11	Stray_FFT_100.0 Hz	FFT, frequency harmonic of 100.0 Hz
12	Vibration__time_reversal_asymmetry_statistic__lag_3	Returns the time reversal asymmetry statistic.
13	Stray_FFT_450.1 Hz	FFT, frequency harmonic of 450.1 Hz
14	Stray__ratio_beyond_r_sigma__r_2	Ratio of values that are more than r * std(x) away from the mean of x.
15	Vibration__augmented_dickey_fuller	The Augmented Dickey-Fuller test is a hypothesis test which checks whether a unit root is present in a time series.
16	Stray_FFT_57.2 Hz	FFT, frequency harmonic of 57.2 Hz
17	Stray_FFT_0.0 Hz	FFT, frequency harmonic of 0.0 Hz
18	Vibration__number_peaks__n_3	Calculates the number of peaks of at least support n in the time series x.
19	Stray_FFT_207.2 Hz	FFT, frequency harmonic of 207.2 Hz
20	Vibration__variation_coefficient	The variation coefficient (standard error / mean) of x

Table 4.16: The 20 most relevant features for the XGBoost classifier from section 4.8. Stray and Vibration in the feature column refers to features originating from the stray field sensors and vibration sensors, respectively.

Rank	Feature	Description
1	Stray_FFT_14.3 Hz	FFT, frequency harmonic of 14.3 Hz
2	Vibration_FFT_114.3 Hz	FFT, frequency harmonic of 114.3 Hz
3	Vibration__partial_autocorrelation__lag_8	The value of the partial autocorrelation function at lag 8
4	Stray_FFT_21.4 Hz	FFT frequency harmonic of 21.4 Hz
5	Vibration__friedrich_coefficients__coeff_3__m_3__r_30	Coefficients of polynomial h(x) , which has been fitted to the deterministic dynamics of Langevin model
6	Vibration__partial_autocorrelation__lag_2	Value of the partial autocorrelation function at lag 2
7	Vibration__partial_autocorrelation__lag_5	Value of the partial autocorrelation function at lag 5
8	Stray_FFT_307.2 Hz	FFT, frequency harmonic of 307.2 Hz
9	Vibration__time_reversal_asymmetry_statistic__lag_3	Returns the time reversal asymmetry statistic.
10	Vibration__quantile__q_0.3	Fixes a corridor given by the quantiles ql and qh of the distribution of x.
11	Vibration__augmented_dickey_fuller	Augmented Dickey-Fuller test is a hypothesis test which checks whether a unit root is present in a time series.
12	Stray_DWT__RWE3	RWE, decomposition level 3
13	Stray_FFT_300.1 Hz	FFT, frequency harmonic of 300.1 Hz
14	Vibration_FFT_0.0 Hz	FFT, DC-component
15	Stray_FFT_85.7 Hz	FFT, frequency harmonic of 85.7 Hz
16	Stray_FFT_7.1 Hz	FFT, frequency harmonic of 7.1 Hz
17	Stray__quantile__q_0.7	Fixes a corridor given by the quantiles ql and qh of the distribution of x.
18	Stray_FFT_100.0 Hz	FFT, frequency harmonic of 100.0 Hz
19	Vibration__abs_energy	Returns the absolute energy of the time series which is the sum over the squared values
20	Vibration__ar_coefficient__coeff_7__k_10	This feature calculator fits the unconditional maximum likelihood of an autoregressive AR(k) process

4.10 Generalizability

Measurements from two industrial generators were made available for this thesis with the purpose of investigating the generalizability of the proposed classifier. Generalizability in this section refers to the ability of classifying samples obtained from generators of which the classifier was not trained on. For confidentiality purposes, the generators are named "Gen1" and "Gen2". Both generators are operational hydropower generators located in Norway. The measurements was obtained to investigate faults in these generators, meaning all measurements from these generators are classified as faulty. Through an analysis of these measurements using advanced signal processing tools it was concluded that "Gen1" suffers from ITSC and dynamic eccentricity, while "Gen2" suffers from static eccentricity of unknown degree [51].

Gen1 is an 8-pole synchronous generator. The measurement series consist of 7 different measurements recorded with 2 different sensor locations for stray magnetic field sensors and vibration sensors. The measurement equipment were the same as that used in this thesis. The different measurements represents different loads and operating characteristics. All signals were measured in steady state.

Gen2 is a 16-pole synchronous generator. The measurement series consists of 2 different measurements recorded at 4 sensor locations. The measurements are for no-load and partial-load, respectively. All signals was recorded at steady state.

The proposed classifier for fault detection was trained on all measurements from the laboratory generator then tested on the measurements obtained from the industrial generators. For Gen1, the logistic regression classifier was trained on dataset containing measurements of SE, while for Gen2 the logistic regression clasifier from section 4.8 was used. The same pre-processing procedure as in section 4.1 was followed for the new measurements, meaning splitting into RSSs and calculating the relevant features. Feature selection was already performed for the laboratory measurements, thus only the features deemed relevant were calculated for the Gen1- and Gen2 measurements. The features was scaled based on the laboratory measurements. The training set, consisting solely of laboratory measurements, was balancing using SMOTE-ENN. As only one class existed for the test set, all classifiers were evaluated based on accuracy. The results are presented in table 4.17. For comparison, SVM with a linear kernel was trained in a similar fashion and produced the same results.

Dataset	Fault	Prediction accuracy
Gen1	ITSC and DE	100%
Gen2	SE	0.3%

Table 4.17: Results from testing the proposed classifier on datasets from industrial hydropower plants.

For detection of ITSC, the classifier correctly classified all samples as faulty. This indicate that the algorithm can detect ITSC across different generators and topologies. For Gen2, close to zero of the samples was correctly classified. It was found that the amplitude of the signals from Gen2 differed from those obtained from the laboratory generator due to the topology of Gen2. Since the amplitude are an important feature for classifying SE, it is likely that the algorithms struggle to identify SE due to the variation in amplitudes.

Chapter 5

Discussion

This chapter presents the discussion of the methodology and results from the previous chapters.

5.1 Measurement Series

One of the objectives of the thesis was to extract measurements of stray magnetic field, vibration, current and voltage. This was performed using the laboratory setup at NTNU and measurement techniques developed in cooperation with Hossein Ehya [51]. For all signals except vibration, the measurement procedures were unproblematic and the signals were of great quality. The vibration measurements was prone to a high amplitude-to-noise ratio. Various methods were tested to eliminate the noise to no avail. The hypothesis was that due to the size of the generator, the vibration was of a to low amplitude to achieve high quality measurements. The laboratory generator is a scaled version of a typical hydropower generator, and due to the scaling factor, the generator created vibrations of a much lower magnitude than of a typical hydropower generator. Based on this, accelerometers of higher sensitivity were purchased, however, due to the late arrival of this equipment it was not fully tested. Further testing of these accelerometers should be carried out to evaluate the hypothesis that the sensitivity of the equipment was the main source of poor measurement quality.

5.1.1 Sensor location

There are 4 large metal bars attached to the stator in the axial direction. These bars are of a magnetic material. The path of the magnetic flux travels through these bars which influences the distribution of the magnetic field in the vicinity of the stator core. This effect is largest in close proximity to the bars. Thus, the measurements are highly affected by the proximity to these metal bars. Based on this it was decided to vary the position of the sensors in axial direction, such that the position in radial direction and the distance to the metal bars remained constant.

For detection of SE, the sensors where placed on the side of the machine with the narrowest airgap, while for detection of ITSC the sensors were placed on the top of the machine. The sensors were placed in three different locations in the axial direction of the machine to avoid the influence of the metal bars. This resulted in sensor location 1 and 3 being relatively close to the edge of the stator, thus being affected by the axial flux at a higher degree. This made the measurements from location 1 and 3 differential from location 2. Sensor location 1 and 3 had to be marked with the same OSSid to avoid target leakage and the different sensor locations in axial direction had only a limited improvement in diversifying the dataset.

The decision to implement several sensor locations came from a need to expand the dataset. It was a necessity to extract more signals of the same machines states to get a representative train and test set. However, if the measurements from different sensor locations are to be used in different datasets, they must be differentiable. Data leakage would occur if the machine learning cannot distinguish between the different sensor locations. In addition, for the test set to be representative, it cannot deviate too much from the training set. If one tests the machine learning algorithms on a test set that are completely different from the training set, the performance would decrease substantially and not reflect the predictive performance of the algorithm. A supervised machine learning algorithm will never be able to predict something outside the problem it is defined on. Thus, it was decided to implement axial direction sensor locations, since it was thought that the axial flux would be enough to distinguish the sensors, but not enough to cause large dissimilarities in the signals.

When experimenting with machine learning it was discovered the majority of sensor locations were not distinguishable and caused data leakage. The sensor locations therefore became more of a problem than a solution, since it increased the number of virtually identical samples in the dataset. It should, based on this result, be analysed if different sensor positions in radial direction could be used in machine learning, as this could solve the problem of both low data and low diversity in the samples. It should be noted that for detection of ITSC, the addition of several sensor locations significantly improved the algorithms. Due to several sensor locations being distinguishable, the training and test set could be made both diverse and representative of most every case measured from the lab generator. This was the core behind the good performance of the proposed algorithm.

5.2 Data Management

One of the largest concerns when working with AI is the amount of data needed to enquire valid results. Since the generator had to be operated in steady state for the feature extraction methods used in this thesis, new measurements taken of previously measured cases would result in effectively identical samples. To exemplify, in this thesis the dataset was highly imbalanced with relatively few samples constituting a healthy machine. To solve this, one could simply measure the healthy case several times, thus increasing the amount of "healthy data", however, these new samples would be approximately identical to the previously measured samples. Thus, measuring several times would in theory be equivalent to duplicating the previous measurements, which are known to cause overfitting without improving the predictive performance. This lack of diversity turned out be a large problem in this thesis. The only way to extract more data was by altering the load or the sensor location. However, these factors have a strict limit to the number of possible combinations. Thus, there exist a ceiling to the dataset, meaning there is a finite limit to the amount of distinguishable samples that are possible to extract from the generator.

In the thesis, 7 different loads were used for the healthy case and 6 loads were used for the faulty cases. This increased the dataset and provided diversity in the measurements. Due to limitations in the converters controlling the power to the generator, the load could not be maintained at an exact value during the measurements. Therefore, the load varied slightly within each measurement. In addition, some of the load increments were quite small. These two factors made it possible that within the measurement of two separate loads, the load was in reality the same. If this were the case, some individual RSSs would create target leakage. This was not seen as a major issue, since it would only be viable for a very small portion of the RSSs, thus not significantly affecting the performance.

All the problems of diversity could be solved by analysing sensor locations. If multiple sensor locations would result in distinguishable measurements, the training and test set could be fully representative of every machine state. It is therefore paramount that further work revolves around data extraction methods. This involves both investigation of the impact of loading in machine learning as well as experimenting with different sensor positions.

5.2.1 Splitting of the OSS

Due to the low amount of data extracted from the laboratory generator it was decided to split the gathered samples into smaller samples called RSS. To further increase the diversity, one electrical period was skipped at each split, resulting in the data set containing the fault signature in various positions within the RSSs. One problem of this methodology is that the generator was operating in stationary conditions during sampling. This results in each of the RSS extracted from the same OSS being relatively similar. This was confirmed when running the algorithm without separating the training-/test set based on OSSid, which achieved a near perfect score on all evaluation metrics. Thus, the splitting of the OSSs could be seen as equivalent to duplicating the measurements. In truth, the dataset for ITSC consisted of 111 measurements duplicated 250 times. One can therefore argue that the splitting of the OSSs were trivial, since duplicating samples usually only results in overfitting. However, this is only true to a certain degree. 111 samples are not nearly enough to fit a machine learning algorithm to a complex task, thus the splitting of the OSSs provided more data for the training of the algorithm at a compromise of the chance of overfitting.

Skipping one electrical period at each split in the OSS was a naive method of diversifying the samples, since the time-series themselves were never used in the machine learning models. When feature engineering were applied to the dataset, the changes imposed by skipping a period were effectively eliminated. The only features affected by the change were a few of the TSFRESH features. The FFT and DWT were entirely unaffected by this adjustment. This was due to the periodicity of the signal. Within each split, regardless of how many electrical periods were skipped, the frequency content and DWT-energy levels remained the same since the periodicity of the signal was kept. Since a large number of the features calculated was unaffected by the split, the small number of affected features was unimportant, as the machine learning would prioritise the unaffected features by assigning larger weights. The affected features would most likely be treated as noise.

5.3 Balancing of the Dataset

It was decided that balancing the classes in the dataset was a necessity. This was due to the low amount of samples representing the healthy case. Initial trials tested without implementation of balancing algorithms resulted in classifiers that had a high score on the faulty samples but an approximately zero detection of healthy samples. The classifiers effectively categorised every sample as faulty. By balancing the dataset, the classifiers were given more input to learn from, thus increasing the detection of the minority case. The SMOTE-ENN algorithm created synthetic samples representing the minority case, then filtered all samples based on a KNN method. This created a more general decision region for the minority class while eliminated outliers and inliers created in the SMOTE process.

When implementing balancing procedures one are effectively saying that the minority class is more important than the class distribution. In a non-laboratory environment, there would most likely be more than enough samples of the healthy state to work with, while fewer to none samples of the faulty state. A balanced dataset was therefore considered more realistic than a dataset with the healthy case representing the minority class. In addition, it is desirable for a power plant to have good detection of all classes to avoid both having a fault go undetected and false alarms. Based on this, balancing the dataset to a 50/50 split of classes was justified.

One problem of implementing resampling methods on the dataset used in this thesis was the large number of identical samples. The splitting of the OSSs into RSSs created 250 near identical samples per OSS. Since the SMOTE-algorithm functions in a stochastic manner, it would create synthetic samples based on samples originating from the same OSS. In such a case the SMOTE-algorithm effectively duplicated samples, since the difference between such samples were zero. This implies that if the dataset that were balanced consisted of a single measurement of the minority class, all synthetic samples from the SMOTE algorithm were duplicates of that class. For the k-fold-CV split this proved an issue, since for certain of the folds the training set contained very few samples of the minority class. For these folds the balancing-methodology only duplicated samples and the classifiers had a poor performance on the minority class. Based on this, the balancing methodology did not improve the performance within the k-fold-CV split, since duplicating samples produce overfitting.

It could also be argued that a simple SMOTE algorithm, i.e. only over-sampling, would be better for this specific dataset. By the addition of under-sampling through ENN, there was a possibility that samples providing a broader scope for the minority class was eliminated. Since there existed a large number of identical samples in the dataset, the ENN algorithm would most likely prioritise these samples over those that differ from the majority. It is therefore a high possibility that the addition of ENN did not aid in detection of the minority class, but rather filtered out samples beneficial for the classification of the minority class. The SMOTE-ENN algorithm was chosen based on a literature search of balancing algorithms and was decided at an early stage as the best fit for this particular dataset. The ideal methodology would have been to test multiple balancing algorithms and evaluate the predictive performance of classifiers on the minority class. At the point of the discovery that the SMOTE-ENN algorithm might not be the best for this dataset, it was not enough time to experiment with different algorithms. Further work should therefore investigate the performance of different balancing methodologies, as the imbalance of the dataset was a prime concern in this thesis.

The balancing methodology was implemented on the training set within each fold in the CV-split. For final evaluation of the algorithm, the SMOTE-ENN algorithm was used on the entire training set. There are conflicting opinions in the machine learning community as to what stage is appropriate for implementation of balancing procedures. Some argue that it should be implemented before feature engineering, i.e. on the time-series themselves, since one cannot predict the influence of the balancing algorithms on the extracted features. Others argue that the implementation should follow the procedure of this thesis.

The argumentation for this procedure is that some of the feature selection methods assume independent features. The independency of features are not kept after implementing SMOTE. In addition, implementing balancing before feature engineering could induce data leakage.

For the dataset in this thesis it was essential that the balancing of classes was done in the latter stages, since the splitting of training and test sets had to be performed according to OSSids to avoid data leakage. This would have been impossible to do after balancing, since SMOTE creates synthetic samples with no accompanying OSSid.

5.4 Training, testing and cross-validation

One of the biggest challenges when working with the data from the laboratory generator was the large number of identical samples and the possible sources of target leakage. Both the individual RSSs, the load and the sensor location was potential sources of target leakage. It therefore quickly became a complicated task to separate the samples into training, testing and cross-validation sets. Several methods were experimented and it was concluded that the only way to properly split the dataset while ensuring no data leakage was by manually selecting the measurements for each set. This method selected sensor location 1 and 3 for training and sensor location 2 for testing. This ensured no data leakage into the test set at the compromise of some data leakage in the CV-folds. It also limited the sampling bias in the test set since all cases was represented. To limit the sampling bias in the cross-validation, a 5-fold CV split was used. As mentioned, there was a high likelihood that certain of these folds contained some target leakage. This methodology could only be applied for the datasets that contained differentiable sensor locations, i.e. dataset A1 and B1 representing ITSC in scenario 1. For the other datasets, either CV-folds based on OSSid or manual selection based splits were used.

There were several key weaknesses to the above mentioned splitting methodology, which was reflected in the results. Due to this another splitting strategy was performed to combat these weaknesses. Instead of splitting by sensor location, which would lead to both overfitting and a relatively different test set, the dataset was split randomly according to OSSids. To avoid target leakage, sensor location 1 and 3 was given the same ID. This proved a much more viable strategy, as both the training and test set consisted of measurements from all sensors. This made the training data more diverse and the test set more representative of the training data. As such the performance of the algorithm significantly increased. Based on these results it is concluded that when dealing with multiple sensor locations, it is essential to ensure that all sensor locations are represented in all sets. It was also concluded that the results using such a split are more reflective of the true fault detective capabilities of the algorithms. The proposed classifier is based on the results using this split and achieved an accuracy and ROC AUC of 0.95.

5.5 Feature Extraction and Importance for Classification

Feature selection through random forest was concluded as the preferred method. These datasets consistently outperformed other selection methods in terms of the performance of the classifiers. Through random forest a significant proportion of the features were eliminated. Since the performance increased by eliminating features from the dataset, it can be concluded that a large portion of the extracted features were redundant.

In the EDA, the DWT features showed a negative correlation to the target label. This is a contradiction to that found in [8]. In addition, only a single DWT feature were included in the most important features for XGBoost and none for Logistic Regression. In [8], the DWT features showed a strong correlation to the target label and was among the most important features for the classifiers. The low value of the DWT features might be due to the implementation of the DWT. In this thesis, all measurements were sampled at a sampling frequency of 10kHz. In [8] measurements were both sampled at 10kHz and 50kHz. The samples were therefore upsampled to 50kHz before feature extraction to ensure the same frequency bands for all samples. This was not performed in this thesis and the frequency bands of the DWT therefore differs between the thesis'. The results indicate that the DWT features proposed in [8] are not robust to variations in sampling frequency and that a sampling frequency of 50kHz should be used for fault detection using the energy of the DWT-levels.

The FFT features and the TSFRESH features were found to be most influential in the logistic regression classifier and the XGBoost classifier. TSFRESH and FFT had approximately paralleled performance, while DWT-features were the least important features among those included. For the stray field signals, FFT-features were the most important. For vibration, the TSFRESH features were the most influential, and paralleled the importance of the stray field features. Both classifiers relied on a variety of features from both vibration and the stray flux. This indicates that sensor-fusion was beneficial in achieving a high performing classifier.

The FFT features from the vibration signals had a low importance in the two classifiers. This is most likely due to the quality of the signal. As mentioned previously, the vibration measurements were prone to noise which affected the FFT-spectrum. Higher quality measurements would probably yield better results in terms of FFT. In addition, FFT-performs poorly on non-stationary signals. One should therefore expect limited results due to the non-stationary nature of vibration. A significant proportion of the features of highest importance for the logistic regression classifier were the FFT features of the stray field. This indicate a strong relationship between the frequency content of stray field and ITSC, confirming the conclusions in [9].

5.6 Machine Learning

The main objective of this thesis was to develop a machine learning algorithm for detection of ITSC and SE. This section discusses the results in light of the objective.

5.6.1 Detection of ITSC

The first attempts at making a classifier split the training and test data based on sensor location. While this ensures every case is reflected in the test set, it provided problems in terms of overfitting and a high variation between the test set and training set. The training data was influenced by the axial flux to a much higher degree. It is therefore unreasonable to expect a high performing classifier using this splitting strategy, since the training and test data differed significantly. This was confirmed in the performance on the test set, which was inadequate.

By implementing a new splitting strategy the performance was significantly increased to a degree potentially viable for industrial application. The proposed classifier boasts a high performance in detection of ITSC. A ROC AUC of 0.95 and specificity of 0.94 on the hold-out data indicates a good ability of distinguishing healthy from faulty with very few false alarms, i.e. false positives. One consequence of the random split is that there was a high likelihood that not all cases were reflected in the test set. *Case* in this sense refers to a specific combination of machine state and load. It was verified that all fault cases were present in the test set, which means the loads were not. Including new measurements of load cases not present in the test set are unlikely to change the performance, since these cases existed in the training set. Since not every case was present in both sets, the high accuracy provides evidence that the classifiers are able to achieve good results on samples not included in the training data. This provides further evidence that the reported performance is valid for new unseen samples.

5.6.2 Detection of Static Eccentricity

For fault detection of SE all classifiers had a near perfect score across all splits. Since feature selection and standardisation could not cause data leakage due the implementation of these methods, there as only two possible causes of leakage: load and sensor location. The OSSids was shared by all sensor locations ensuring that they would end up in the same dataset. This way, if one measurement from sensor location 1 was in the training set, the corresponding measurements from sensor location 2 and 3 was also in the training set. This made data leakage due to sensor location impossible.

Depending on the influence of loading on the measured signal, a small variation in loading could result in effectively identical measurements. Target leakage could then occur if the load increments were too small. This would however, also affect the results using datasets of ITSC, since the load increments were the same. Since there was no significant target leakage due to loading in the results using dataset A1 and B1, the logical conclusion is that the loading was not a source of target leakage. In addition, several combinations of loads were tested for both the training-, testing- and CV-set. If load was a cause for target leakage it is highly likely that this would have been detected across these trials. Based on the testing of different splits and the results from the ITSC-datasets, it was concluded that the results were not due to target leakage.

When static eccentricity occurs in the machine the magnetic field distribution changes. The strength of the magnetic field at the side with the shortest airgap is effectively increased due to the eccentricity. There should in theory exist a distinct relationship between the degree of SE and the amplitude of the measured signals. The perfect score of the classifiers could be due to the algorithms responding to such a relationship based on the time-series features extracted by TSFRESH. Since the possible sources for target leakage had been exhausted it was concluded that the perfect score reflects the classifiers true ability of detecting static eccentricity.

5.6.3 Impact of load

The analysis of the impact of load concluded that the loading of the generator had a impact on the performance of the machine learning algorithm. By eliminating the no-load measurements the performance increased with approximately 3%. This shows that to achieve the highest performance within fault detection of ITSC, the analysis should be performed on a loaded generator.

The vibration signal in the generator at no-load is very limited. Since there is no current in the stator windings during no-load, the vibration in the machine is significantly lower. This means that for the no-load measurements, the algorithm relied mostly on features from the stray field. As such, it is expected a lower performance on the no-load samples. By omitting the no-load measurements, the performance increase since the algorithms can rely on features from both stray field and vibration for all samples included in the dataset.

5.6.4 Validity of performance

Since the test data used to evaluate the performance of the proposed algorithm was extracted at the very beginning of the development procedure, target leakage into this set was effectively eliminated. However, as was seen in the process of developing the algorithm, there was a relatively large sampling bias. The sampling bias was naturally largest for the cases with the lowest amount of data, referring to SE and scenario 2, but there is a high likelihood that sampling bias was also present in the datasets for ITSC. For this reason it is naive to expect the same performance on new samples.

5.6.5 Generalizability

One of the objectives of the thesis was to test the performance of the algorithm using measurements from industrial generators. The purpose of this analysis was to evaluate the possibility of a generalizable algorithm. Applying a machine learning algorithm to a new domain rarely works well due to differences in the domains. Machine learning are often very specialized at solving the specific task it was designed to solve.

The datasets obtained from Gen1 and Gen2 contained only one class, and Gen1 suffered from multiple faults. In addition, the measuring techniques differed, and multiple sensor locations were used. As seen in this thesis, sensor location is vital when performing fault detection. These conditions should be taken into considerations when evaluating the results.

The algorithm could not detect SE in Gen2. The characteristics of Gen2 resulted in a signal of a different amplitude than that measured in the laboratory generator. Since detection of SE relies heavily on the time-domain features such as the amplitude, the failure in detection of SE reflects the differences in the signals. This shows the problems faced when applying machine learning to new domains. The algorithm was fitted to a specific generator and has limited capabilities of classifying samples originating from a different generator when it comes to static eccentricity.

The high accuracy in detection of ITSC in Gen1 demonstrates how an algorithm can be trained in a laboratory environment and achieve high results in an industrial environment. It also shows that the algorithm can detect faults across multiple sensor locations and generator topologies.

By using advanced signal processing techniques it was concluded that Gen1 suffers from ITSC and DE [51]. It was found that the generator topology and power rating did not significantly affect the features related to ITSC. This result justifies the accuracy obtained by the algorithm. Since it is possible to detect the fault using signal processing tools, irrespective of the nature of the measurements or the generator characteristics, the algorithm should detect the fault at a high accuracy.

Transfer Learning

Transfer learning might be a solution to the problem of generalization. In transfer learning, a machine learning model is first made based on the laboratory generator. This model is then used as basis for a new machine learning model for a separate generator. The "old model" is built upon by combining more measurements from the new generator. The results from Gen1 indicate that the proposed algorithm can detect faulty cases in a different domain. By incorporating healthy data from this new domain in the transfer learning, it might be possible to achieve a high performing classifier for detection of faults across multiple different generators. Healthy data are easily extracted in an industrial environment, while faulty samples not so much. Transfer learning can circumvent this problem by using the faulty and healthy samples from the laboratory generator as basis for detecting faulty samples in a different generator. Thus, the algorithm would not be dependant on faulty measurements from the new domain. This could increase the detection rate of static eccentricity across domains as well. By combining healthy data from a separate SPSG into machine learning model, the variety in the measurement would induce new decision boundaries that could increase the predictive performance across domains.

5.7 Contribution of Multi-Sensor Fusion

When discussing the contribution of sensor-fusion it is natural to compare the results with those obtained in [8]. However, these results are not directly comparable due to the difference in class balance between these thesis'. Any classifier that performs reasonably well on the positive class, i.e. the faulty case, would boast a high accuracy, precision, recall and F1-score on the dataset used in this thesis, which makes comparing these metrics trivial.

Another condition that complicates comparisons are the fact that this thesis had significantly more data to work with. Due to the low amount of data in the [8], the results were prone to a large sampling bias. The reported performance of the preceding thesis is therefore likely to change if a new split of training and testing data were used. This makes the results questionable, since it is unclear if they reflect the true performance of the algorithms or if they are a function of that exact split. In [8], 15% of the available data were used as hold out data. This is equivalent to 7 measurements. The thesis had a total of 9 different cases, not accounting for loading. In addition, the split was performed in a random fashion, thus it is likely that certain cases was reflected more than once in the hold-out set. As such, there was no possible split that could ensure all fault cases were tested, and most likely only a fraction of the cases was actually in the hold-out set. The argument is therefore that the reported performance in [8] was not actually reflective of the capabilities of the algorithm, since only a fraction of the cases were tested. The sampling bias makes the results in [8] highly questionable, and the performance is found likely to change if the algorithm was tested on more data.

The ROC AUC score provides for some comparison since it is tied to both the positive and the negative class. The ROC AUC quantifies the classifiers ability to distinguish the classes from each other and has been proven in the literature to be a powerful metric for comparing machine learning algorithms. In addition, the ability to separate classes is a desired characteristic for an algorithm intended for industrial application, since it would decrease false alarms and ensure that faulty states are detected. The ROC AUC of the proposed classifier was approximately 0.95, while the highest performing classifier in [8] was about 0.84. This demonstrates a significantly better ability to distinguish classes.

When taking in all these factors and evaluating the algorithms based on a suited evaluation metric, it is clear that the capabilities of fault detection has been improved in the proposed classifier. The increased performance of the proposed algorithm were a combination of more data and sensor-fusion. It is concluded that sensor-fusion contributed in the increased predictive performance since the algorithms utilized features from all sensors. The performance reflects the quality of the features, and a good performance signals few redundant features and a strong relationship between features and target label.

Scenario 2 was designed for the purpose of evaluating the contribution of sensor-fusion in fault detection. Unfortunately, the dataset had to be limited to a very few set of samples and was therefore eliminated from further analysis. If a classifier was made based on this dataset, it would have an enormous sampling bias. It was therefore deemed unreasonable to proceed with this scenario as the results would at best be questionable. Based on this it should be investigated if distinguishable measurements of current and voltage can be extracted. One possibility is to measure the phase current/voltage of each phase in the machine. This was not investigated in this thesis.

Chapter 6

Conclusion

This thesis investigates the usage of machine learning in combination with sensor-fusion for the detection of Inter-Turn Short-Circuit (ITSC) and Static Eccentricity (SE) in synchronous generators. This was performed using measurements from a laboratory generator at NTNU for the training and testing of different machine learning models. In addition, the generalizability of the proposed algorithm has been tested using measurements obtained from two industrial hydropower generators.

The laboratory setup consisted of a 14-pole, 100kVA synchronous generator powered by a 90kW induction machine. The generator was design to reflect the properties of a typical hydropower generator, only scaled down for practical purposes. The generator could be imposed with both ITSC and static eccentricity. Several tests on the 100 kVA SPSG were performed to measure the stray magnetic flux, vibration, stator current and stator voltage as a part of the thesis. The measurements included 6 different loads and three different sensor locations.

The sensor locations of the stray field and vibration sensors was varied in the axial direction of the machine. This was done to limit the impact of sensor location in the measurements while ensuring diversity in the dataset used for machine learning. It was later discovered that most of the sensor locations in axial direction was not distinguishable in the eyes of the machine learning, thereby causing target leakage. It was based on this concluded that for the purpose of fault detection using machine learning, several sensor locations in radial direction of the generator should be investigated.

Vibration measurements proved troublesome to extract to a satisfactory quality. The main problem with the vibration measurements was a low amplitude to noise ratio. This was most likely due to the size of the generator and the sensitivity of the equipment. Further testing with equipment of higher sensitivity should be pursued to increase the quality of the vibration measurements.

Voltage and current were used in sensor fusion with the stray field and vibration measurements. Due to the inability to change sensor location of the current and voltage sensor, the size of the dataset was severely limited. Only one of the three sensor locations used for stray field and vibration could be used in combination with voltage and current. This was reflected in the results from the machine learning, which was sub-optimal. Further efforts to make sensor-fusion with all signals usable in machine learning was dropped since the size of the dataset was too small. It should be investigated into methods of obtaining distinguishable measurements of voltage and current for the purpose of increasing the size of the dataset. As such, only sensor-fusion of stray flux and vibration was pursued for fault detection.

Features calculated from the extracted time-series was used as input in the machine learning. Features were calculated by FFT, DWT and TSFRESH. The FFT and TSFRESH features paralleled in performance and outperformed that of the DWT features. A significant proportion of the features of highest importance for the logistic regression classifier were the FFT features of the stray field. This indicates a strong relationship between the frequency content of stray field and ITSC. Features from both stray field and vibration were among the most important features, indicating that sensor-fusion was beneficial in fault detection.

The features were filtered by means of a Random Forest classifier and the TSFRESH algorithm for their relevance in classifying ITSC and SE. Random Forest filtered out a significantly higher number of features. The classifiers tested and trained on the dataset filtered through Random forest consistently outperformed those trained on TSFRESH-filtered data. As such, Random forest is preferred both due to performance of the classifiers and due to the dimensions of the feature space.

The dataset was initially split into training and hold-out set based on sensor location. The classifiers were then optimized using a grid-search of the most relevant parameters for each classifier. This resulted in classifiers that overfit the training data and performed poorly on the hold-out data. The highest-ranking classifier had a ROC AUC of 0.75. It was concluded that the poor results were a function of the sub-optimal splitting strategy. All sensor locations should have been represented in both the training set and the hold-out-set. To remedy this poor results a new splitting strategy was implemented.

Through splitting the data by OSSids using a stochastic process, the fault detective capabilities of the classifiers improved significantly. The highest scoring classifier were logistic regression, with and accuracy of 95% and a ROC AUC of 0.95 on the hold-out-data. This confirms that when dealing with multiple sensor locations, it is essential that all locations are reflected in both the training and the testing data. The sensor-fusion of stray field and vibration was found to be a contributing factor to the high performance. The classifiers detected SE with a 100% accuracy. It was concluded that this was not a result of target leakage.

The generalizability of the algorithm to new domains was evaluated by testing the proposed algorithm on measurements obtained from industrial hydropower plants. The classifier detected ITSC with an accuracy of 100% and static eccentricity with and accuracy of 0%. This shows that the algorithm is able to detect ITSC across multiple generator topologies. Detection of static eccentricity relied heavily on the amplitude of the signals. The difference in amplitude of the generators were concluded to be the source of the bad performance.

6.1 Further work

- Investigate methods of diversifying the dataset. This should include an analysis of the effect of sensor displacement in the radial axis of the generator.
- Investigate methods of improving the quality of the vibration measurements. An evaluation of the relevance of FFT features extracted from vibration measurements should also be conducted.
- Multi-sensor fusion by non-linearly averaging the different signals, Dempster-Shafer evidence theory or other fusion techniques to improve performance beyond that obtained in this thesis.
- Investigate into anomaly detection algorithm for use in the hydropower industry. Anomaly detection algorithms are trained on one class with the task of detecting anomalies. This could circumvent the challenges of extracting measurements of faulty generators and are a possible solution to on-line condition monitoring for industrial application.
- Investigate transfer learning based on the proposed algorithm for use on industrial generators. Transfer learning are likely to be more successful using more advanced machine learning than that presented in this thesis.

Bibliography

[1] NVE, “Kraftproduksjon,” (accessed: 01.12.2020). [Online]. Available: https://www.nve.no/energiforsyning/kraftproduksjon/?ref=mainmenu

[2] Electric Power Reasearch Institute, “Main generator on line monitoring and diagnostics,” EPRI, Tech. Rep., 1996.

[3] M. Mohamed, E. Mohamed, A. Mohamed, M. Abdel-Nasser, and M. M. Hassan, “Detection of inter turn short circuit faults in induction motor using artificial neural network,” in *2020 26th Conference of Open Innovations Association (FRUCT)*, 2020, pp. 297–304.

[4] K. Kudelina, T. Vaimann, B. Asad, A. Rassõlkin, A. Kallaste, and G. Demidova, “Trends and challenges in intelligent condition monitoring of electrical machines using machine learning,” *Applied Sciences*, vol. 11, no. 6, 2021. [Online]. Available: https://www.mdpi.com/2076-3417/11/6/2761

[5] A. N. Saberi, S. Sandirasegaram, A. Belahcen, T. Vaimann, and J. Sobra, “Multi-sensor fault diagnosis of induction motors using random forests and support vector machine,” in *2020 International Conference on Electrical Machines (ICEM)*, vol. 1, 2020, pp. 1404–1410.

[6] R. Gogulaanand, T. Balasubramaniyavijayan, R. Arunsivaram, S. Aishwarya, P. V. S. Nag, and C. S. Kumar, “Intelligent monitoring of synchronous generators in smart grids using deep neural network,” in *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, 2019, pp. 1376–1379.

[7] M. Rahnama, A. Vahedi, A. M. Alikhani, and A. Montazeri, “Machine-learning approach for fault detection in brushless synchronous generator using vibration signals,” *IET Science, Measurement Technology*, vol. 13, no. 6, pp. 852–861, 2019.

[8] T. N. Skreien, “Application of signal processing and machine learning tools in fault detection of synchronous generators,” Master’s thesis, NTNU, Department of Electric Power Engineering, 06 2020.

[9] O. Sørheim, “Fault detection of inter-turn short-circuit in synchronous generator using stray magnetic flux,” Department of Electrical Power Engineering, NTNU – Norwegian University of Science and Technology, Project report in TET4520, 12 2020.

[10] G. Rødal, “Online condition monitoring of synchronous generators using vibration signal,” Master’s thesis, NTNU, Department of Electric Power Engineering, 06 2020.

[11] R. C. Luo, Chih-Chen Yih, and Kuo Lan Su, “Multisensor fusion and integration: approaches, applications, and future research directions,” *IEEE Sensors Journal*, vol. 2, no. 2, pp. 107–119, 2002.

[12] B. Chandrasekaran, S. Gangadhar, and J. M. Conrad, “A survey of multisensor fusion techniques, architectures and methodologies,” in *SoutheastCon 2017*, 2017, pp. 1–8.

[13] T. P. Banerjee and S. Das, “Multi-sensor data fusion using support vector machine for motor fault detection,” *Information Sciences*, vol. 217, pp. 96–107, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025512004185

[14] R. Razavi-Far, E. Hallaji, M. Farajzadeh-Zanjani, M. Saif, S. H. Kia, H. Henao, and G.-A. Capolino, “Information fusion and semi-supervised deep learning scheme for diagnosing gear faults in induction machine systems,” *IEEE Transactions on Industrial Electronics*, vol. 66, no. 8, pp. 6331–6342, 2019.

[15] S. Ma, Y. Yuan, J. Wu, Y. Jiang, B. Jia, and W. Li, “Multisensor decision approach for hvcb fault detection based on the vibration information,” *IEEE Sensors Journal*, vol. 21, no. 2, pp. 985–994, 2021.

[16] J. Faiz, B. M. Ebrahimi, and H. A. Toliyat, “Effect of magnetic saturation on static and mixed eccentricity fault diagnosis in induction motor,” *IEEE Transactions on Magnetics*, vol. 45, no. 8, pp. 3137–3144, 2009.

[17] H. Ehya and J. Faiz, “Dynamic and static eccentricity fault diagnosis in salient-pole synchronous generator using time stepping finite elements method,” in *IEEE International Magnetics Conference, INTERMAG Europe 2014, Dresden*, 05 2014.

[18] H. Ehya, I. Sadeghi, and J. Faiz, “Eccentricity fault indices in large induction motors an overview,” in *2017 8th Power Electronics, Drive Systems Technologies Conference (PEDSTC)*, 2017, pp. 329–334.

[19] ——, “Online condition monitoring of large synchronous generator under eccentricity fault,” in *2017 12th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, 2017, pp. 19–24.

[20] J. Antonino-Daviu, P. Rodriguez, M. Riera-Guasp, A. Arkkio, J. Roger-Folch, and R. Pérez, “Transient detection of eccentricity-related components in induction motors through the hilbert–huang transform,” *Energy Conversion and Management*, vol. 50, pp. 1810–1820, 07 2009.

[21] G. C. Stone, M. Sasic, J. Stein, and C. Stinson, “Using magnetic flux monitoring to detect synchronous machine rotor winding shorts,” in *2011 Record of Conference Papers Industry Applications Society 58th Annual IEEE Petroleum and Chemical Industry Conference (PCIC)*, 2011, pp. 1–7.

[22] H. Ehya, I. Sadeghi, J. Faiz, and A. A. S. Akmal, “Online condition monitoring of large synchronous generator under short circuit fault — a review,” in *2018 IEEE International Conference on Industrial Technology (ICIT)*, 2018, pp. 1843–1848.

[23] I. L. Groth, “On-line magnetic flux monitoring and incipient fault detection in hydropower generators,” Master’s thesis, NTNU, Department of Electric Power Engineering, 05 2019.

[24] A. Ng, “Machine learning,” [MOOC] Coursera, (accessed: 25.02.2021). [Online]. Available: https://www.coursera.org/learn/machine-learning

[25] S. Badillo, B. Banfai, F. Birzele, I. Davydov, L. Hutchinson, T. Kam-Thong, J. Siebourg-Polster, B. Steiert, and J. D. Zhang, “An introduction to machine learning,” *Clinical Pharmacology Therapeutics*, vol. 107, 03 2020.

[26] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: with Applications in R.* Springer, 2013.

[27] T. M. Mitchell, *Machine Learning.* McGraw Hill, 1997.

[28] A. Navada, A. N. Ansari, S. Patil, and B. A. Sonkamble, "Overview of use of decision tree algorithms in machine learning," in *2011 IEEE Control and System Graduate Research Colloquium*, 2011, pp. 37–42.

[29] K. J. Archer and R. V. Kimes, "Empirical characterization of random forest variable importance measures," *Computational Statistics Data Analysis*, vol. 52, no. 4, pp. 2249–2260, 2008.

[30] R. Sakata, I. Ohama, and T. Taniguchi, "An extension of gradient boosted decision tree incorporating statistical tests," in *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, 2018, pp. 964–969.

[31] C. Bentéjac, A. Csörgő, and G. Martínez-Muñoz, "A comparative analysis of xgboost," 11 2019.

[32] S. Adebayo, "How the kaggle winners algorithm xgboost works," (accessed: 30.04.2021). [Online]. Available: https://dataaspirant.com/xgboost-algorithm/

[33] J. Bao, "Multi-features based arrhythmia diagnosis algorithm using xgboost," in *2020 International Conference on Computing and Data Science (CDS)*, 2020, pp. 454–457.

[34] S. Kiliçarslan and M. Celik, "Rsigelu: A nonlinear activation function for deep neural networks," *Expert Systems with Applications*, vol. 174, p. 114805, 2021.

[35] Y. Koçak and G. Üstündağ Şiray, "New activation functions for single layer feedforward neural network," *Expert Systems with Applications*, vol. 164, p. 113977, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417420307557

[36] S. Langer, "Approximating smooth functions by deep neural networks with sigmoid activation function," *Journal of Multivariate Analysis*, vol. 182, p. 104696, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0047259X20302773

[37] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* New York, NY, USA: Association for Computing Machinery, 2016. [Online]. Available: https://doi.org/10.1145/2939672.2939785

[38] H. Kadkhodaei and A. M. E. Moghadam, "An entropy based approach to find the best combination of the base classifiers in ensemble classifiers based on stack generalization," in *2016 4th International Conference on Control, Instrumentation, and Automation (ICCIA)*, 2016, pp. 425–429.

[39] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *Journal of Machine Learning Research*, vol. 18, no. 17, pp. 1–5, 2017. [Online]. Available: http://jmlr.org/papers/v18/16-365.html

[40] N. Chawla, K. Bowyer, L. Hall, and W. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *J. Artif. Intell. Res. (JAIR)*, vol. 16, pp. 321–357, 06 2002.

[41] M. Hossin and M. Sulaiman, "A review on evaluation metrics for data classification evaluations," *International Journal of Data Mining & Knowledge Management Process*, vol. 5, no. 2, p. 1, 2015.

[42] E. Keedwell, "An analysis of the area under the roc curve and its use as a metric for comparing clinical scorecards," in *2014 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 2014, pp. 24–29.

[43] V. Tavşanoğlu, "Teaching aliasing and spectral leakage through the sampling of images," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2019, pp. 1–5.

[44] M. Molinas and L. A. Moctezuma, "Lecture notes in Adaptive Data Analysis - Theory and Application," NTNU – Norwegian University of Science and Technology, 2020.

[45] M. López, M. Molinas, and G. Kulia, "Understanding instantaneous frequency detection: A discussion of hilbert-huang transform versus wavelet transform," 09 2017.

[46] M. Bahoura and J. Rouat, "Wavelet speech enhancement based on the teager energy operator," *IEEE Signal Processing Letters*, vol. 8, no. 1, pp. 10–12, 2001.

[47] L. Guo, D. Rivero, J. A. Seoane, and A. Pazos, "Classification of eeg signals using relative wavelet energy and artificial neural networks," 01 2009, pp. 177–184.

[48] M. Christ, N. Braun, and J. Neuffer, "Time series feature extraction," (accessed: 25.02.2021). [Online]. Available: https://tsfresh.readthedocs.io/en/latest/text/introduction.html

[49] Y. Benjamini and D. Yekutieli, "The control of the false discovery rate in multiple testing under dependency," *Ann. Stat.*, vol. 29, 08 2001.

[50] D. C. Hanselman, *Brushless Permanent Magnet Motor Design*, 2nd ed. Magna Physics Publishing, 2006.

[51] H. Ehya, "Private conversation," Doctoral Research Fellow at the Department of Electrical Engineering, NTNU, Spring 2020.

Appendices

Appendix A

Measurements

A.1 Measurement Series

Table A.1: Measurement series from the experimental work. Each measurement was sampled for all individual signals (stray field, vibration, current and voltage). The loading refers to the cases explained in table 3.3. All experiments were performed with three sensor locations, except for voltage and current where one sensor location was used.

Measurement	Case	Sampling frequency	Loading
1	Healthy	10kHz	NL
2	Healthy	10kHz	L1
3	Healthy	10kHz	L2
4	Healthy	10kHz	L3
5	Healthy	10kHz	L4
6	Healthy	10kHz	L5
7	Healthy	10kHz	L6
8	1 turn SC	10kHz	NL
9	1 turn SC	10kHz	L1
10	1 turn SC	10kHz	L2
11	1 turn SC	10kHz	L3
12	1 turn SC	10kHz	L4
13	1 turn SC	10kHz	L5
14	2 turn SC	10kHz	NL
15	2 turn SC	10kHz	L1
16	2 turn SC	10kHz	L2
17	2 turn SC	10kHz	L3
18	2 turn SC	10kHz	L4
19	2 turn SC	10kHz	L5
20	3 turn SC	10kHz	NL
21	3 turn SC	10kHz	L1
22	3 turn SC	10kHz	L2
23	3 turn SC	10kHz	L3
24	3 turn SC	10kHz	L4
25	3 turn SC	10kHz	L5
26	7 turn SC	10kHz	NL

Table A.1 continued from previous page

Measurement	Case	Sample Rate	Loading
27	7 turn SC	10kHz	L1
28	7 turn SC	10kHz	L2
29	7 turn SC	10kHz	L3
30	7 turn SC	10kHz	L4
31	7 turn SC	10kHz	L5
32	10 turn SC	10kHz	NL
33	10 turn SC	10kHz	L1
34	10 turn SC	10kHz	L2
35	10 turn SC	10kHz	L3
36	10 turn SC	10kHz	L4
37	10 turn SC	10kHz	L5
38	10% SE	10kHz	NL
39	10% SE	10kHz	L1
40	10% SE	10kHz	L2
41	10% SE	10kHz	L3
42	10% SE	10kHz	L4
43	10% SE	10kHz	L5
44	20% SE	10kHz	NL
45	20% SE	10kHz	L1
46	20% SE	10kHz	L2
47	20% SE	10kHz	L3
48	20% SE	10kHz	L4
49	20% SE	10kHz	L5

Appendix B

Exploratory Data Analysis

B.1 EDA of the dataset for static eccentricity

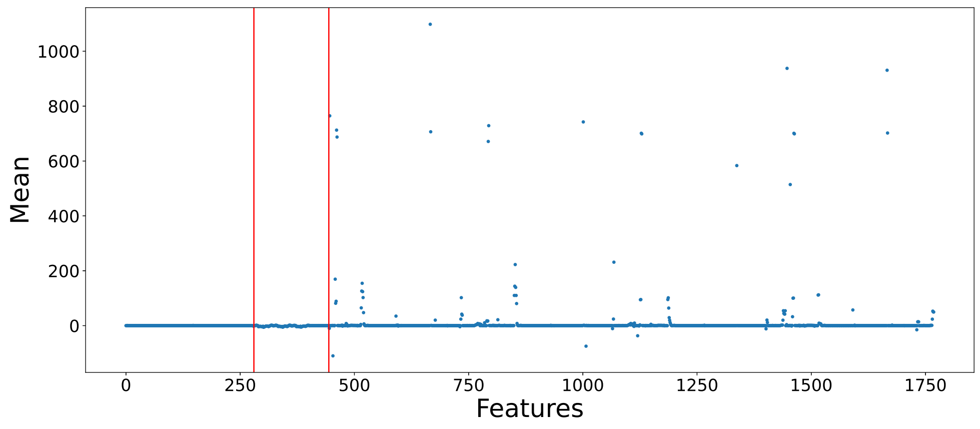


Figure B.1: Mean of all individual features in the dataset for SE. The red lines separate the section representing features from FFT, DWT and TSFRESH, from left to right. Inside each of these section the features originating from stray magnetic field, vibration, current and voltage are in the respective order from left to right.

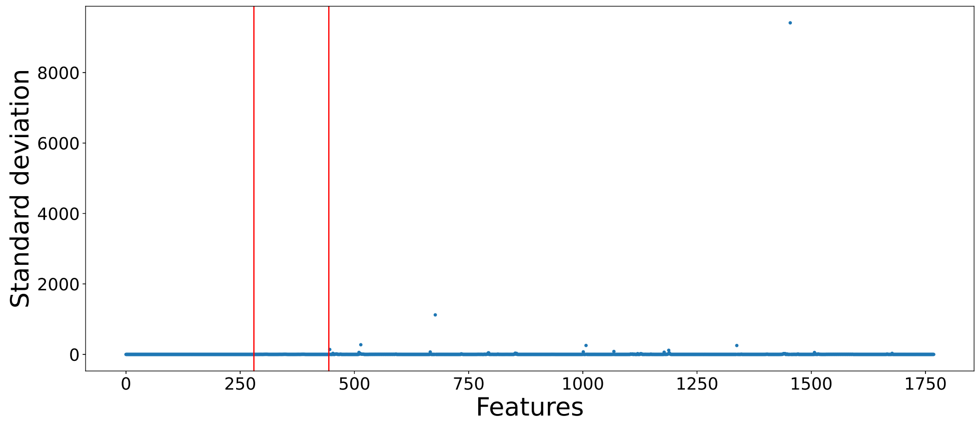


Figure B.2: Standard deviation of all individual features in the dataset for SE. The red lines separate the section representing features from FFT, DWT and TSFRESH, from left to right. Inside each of these section the features originating from stray magnetic field, vibration, current and voltage are in the respective order from left to right.

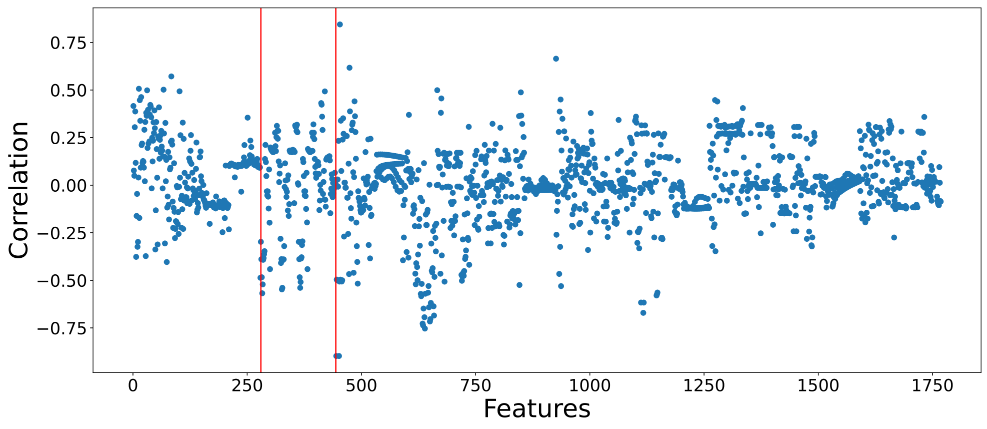


Figure B.3: Correlation of individual features to the target label in the dataset for SE. The red lines separate the section representing features from FFT, DWT and TSFRESH, from left to right. Inside each of these section the features originating from stray magnetic field, vibration, current and voltage are in the respective order from left to right.

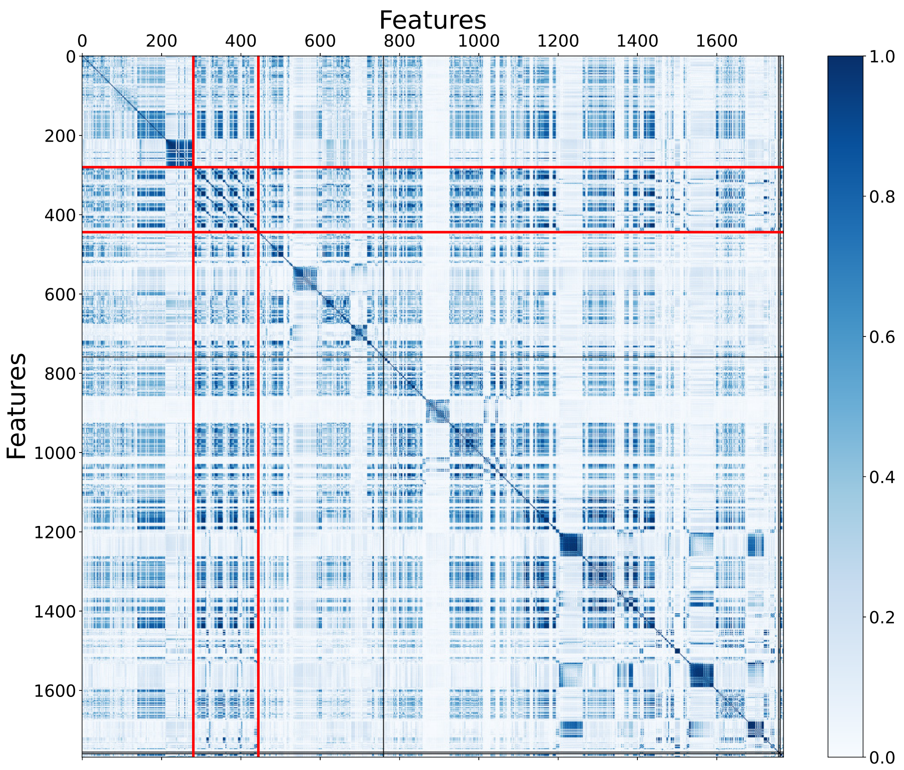


Figure B.4: Correlation matrix for the dataset containing measurements of SE. The red lines represents the transition from FFT-, DWT- and TSFRESH-features from left to right.

B.2 The 20 highest correlated features for each dataset

Table B.1: The 20 features with the highest correlation to ITSC from sensor-fusion scenario 1.

Features	Correlation
Stray: Permutation entropy (τ = 1, D = 7)	0.4535
Vibration: Ratio of values that are more than r * std(x) (r=0.5)	0.4490
Stray: Permutation entropy (τ = 1, D = 6)	0.4441
Vibration: Aggregated autocorrelation (f_{agg} = median, maxlag = 40)	0.4418
Stray: Number of peaks greater than n=1 neighbours	0.4348
Stray: Permutation entropy (τ = 1, D = 3)	0.4324
Stray: Permutation entropy (τ = 1, D = 5)	0.4255
Stray: unconditional maximum likelihood of an autoregressive process (coeff = 10, k=10)	0.4254
Stray: FFT component (278.6 Hz)	0.4250
Vibration: Partial autocorrelation (lag = 5)	0.4202
Vibration: Autocorrelation (lag = 7)	0.4201
Stray: Permutation entropy (τ = 1, D = 4)	0.4181
Vibration: Autocorrelation (lag = 6)	0.4180
Stray: unconditional maximum likelihood of an autoregressive process (coeff = 9, k=10)	0.4131
Vibration: Autocorrelation (lag = 8)	0.4049
Stray: FFT component (250.0 Hz)	0.4024
Vibration: Autocorrelation (lag = 5)	0.3984
Stray: Number of peaks greater than n=3 neighbours	0.3904
Vibration: Longest strike above mean	0.3777
Stray: FFT component (264.3 Hz)	0.3765

Table B.2: The 20 features with the highest correlation to SE from sensor-fusion scenario 1.

Features	Correlation
Stray: Variation coefficient	0.8924
Stray: Mean	0.8637
Stray: Sum of all values	0.8637
Vibration: Cross power spectral density, coeff 2	0.5595
Stray: HWE decomposition level 4	0.5235
Stray: IWE decomposition level 4	0.5214
Vibration: IWE decomposition level 4	0.5144
Stray: The count of each value in interval [-1,1]	0.5141
Stray: HWE decomposition level 5	0.5109
Stray: IWE decomposition level 5	0.4984
Stray: linear least-squares regression, chunk_len = 5, f_agg ="max"	0.4968
Stray: linear least-squares regression, chunk_len = 10, f_agg = "max"	0.4950
Stray: quantile (q=0.9)	0.4933
Vibration: IWE decomposition level 5	0.4919
Stray linear least-squares regression	0.4851
Stray: Standard deviation	0.4851
Stray Root mean square	0.4849
Stray: Maximum value	0.4848
Vibration: FFT component (100.0 Hz)	0.4810
Stray: Linear least-squares regression, chunk_len=5, f_agg="mean"	0.4808

Table B.3: The 20 features with the highest correlation to ITSC from sensor-fusion scenario 2.

Features	Correlation
Stray: Variation coefficient (standard error/mean)	0.8331
Stray: Mean of quantile [0.4, 1], absolute value = True	0.6996
Stray: Mean of quantile [0.4, 0.8], absolute value = True	0.6924
Stray: Variance of quantile [0.4, 0.8], absolute value = False	0.6893
Stray: Variance of quantile [0.4, 1], absolute value = False	0.6820
Stray: Mean of quantile [0.6, 1], absolute value = True	0.6693
Stray: Mean of quantile [0.2, 0.8], absolute value = True	0.6660
Stray: Variance of quantile [0.6, 1], absolute value = False	0.6520
Stray: Mean of quantile [0.2, 1], absolute value = True	0.6425
Stray: Mean of quantile [0.6, 0.8], absolute value = True	0.6414
Stray: Variance of quantile [0.4, 0.8], absolute value = True	0.6363
Stray: Standard deviation	0.6315
Stray: Linear least-squares regression	0.6314
Stray: Root mean square	0.6311
Stray: Mean of quantile [0.2, 0.6]	0.6307
Stray: Linear least-squares regression, chunk_len=5, f_agg="max"	0.6253
Stray: Linear least-squares regression, chunk_len=5, f_agg="mean"	0.6248
Stray: Variance of quantile [0.2, 0.8], absolute value = False	0.6230
Stray: Variance of quantile [0.6, 0.8], absolute value = False	0.6207
Stray: Linear least-squares regression, chunk_len=10, f_agg="mean"	0.6202

Table B.4: The 20 features with the highest correlation to SE from sensor-fusion scenario 2.

Features	Correlation
Stray: Mean	0.8975
Stray: Sum of all values	0.8975
Stray: Variation coefficient	0.8445
Stray: Mean of quantile [0.2, 1], absolute value = True	0.7533
Stray: Mean of quantile [0.2, 0.8], absolute value = True	0.7366
Stray: Variance of quantile [0.2, 0.8], absolute value = False	0.7274
Stray: Variance of quantile [0.4, 1], absolute value = False	0.7162
Stray: Mean of quantile [0.4, 1], absolute value = True	0.7038
Stray: Variance of quantile [0.2, 1], absolute value = False	0.6938
Stray: Mean of quantile [0.6, 1], absolute value = True	0.6849
Current: Median	0.6707
Vibration: Cross power spectral density, coeff 2	0.6644
Stray: Variance of quantile [0.2, 0.8], absolute value = True	0.6482
Stray: Variance of quantile [0.4, 0.8], absolute value = True	0.6413
Stray: Variance of quantile [0.6, 1], absolute value = False	0.6364
Stray: Variance of quantile [0.4, 1], absolute value = True	0.6187
Stray: Minimum value	0.6170
Current: Mean	0.6168
Current: Sum of all values	0.6168
Stray: Mean of quantile [0.2, 0.6], absolute value = True	0.5832

Appendix C

Machine learning

C.1 Feature Engineering

Table C.1: Description of features calculated by TSFRESH [48]

Features	Description
abs_energy(x)	The absolute energy of the time series which is the sum over the squared values
absolute_maximum(x)	Calculates the highest absolute value of the time series x.
absolute_sum_of_changes(x)	Returns the sum over the absolute value of consecutive changes in the series x
agg_autocorrelation(x, param)	Descriptive statistics on the autocorrelation of the time series.
agg_linear_trend(x, param)	Calculates a linear least-squares regression for values of the time series that were aggregated over chunks versus the sequence from 0 up to the number of chunks minus one.
approximate_entropy(x, m, r)	Implements a vectorized Approximate entropy algorithm.
ar_coefficient(x, param)	Fits the unconditional maximum likelihood of an autoregressive AR(k) process.
augmented_dickey_fuller(x, param)	Does the time series have a unit root?
autocorrelation(x, lag)	Calculates the autocorrelation of the specified lag, according to the formula
benford_correlation(x)	Useful for anomaly detection applications.
binned_entropy(x, max_bins)	First bins the values of x into max_bins equidistant bins.
c3(x, lag)	Uses c3 statistics to measure non linearity in the time series
change_quantiles(x, ql, qh, isabs, f_agg)	First fixes a corridor given by the quantiles ql and qh of the distribution of x.
cid_ce(x, normalize)	This function calculator is an estimate for a time series complexity
count_above(x, t)	Returns the percentage of values in x that are higher than t
count_above_mean(x)	Returns the number of values in x that are higher than the mean of x
count_below(x, t)	Returns the percentage of values in x that are lower than t
count_below_mean(x)	Returns the number of values in x that are lower than the mean of x
cwt_coefficients(x, param)	Calculates a Continuous wavelet transform for the Ricker wavelet
energy_ratio_by_chunks(x, param)	Calculates the sum of squares of chunk i out of N chunks expressed as a ratio with the sum of squares over the whole series.
first_location_of_maximum(x)	Returns the first location of the maximum value of x.
first_location_of_minimum(x)	Returns the first location of the minimal value of x.
fourier_entropy(x, bins)	Calculate the binned entropy of the power spectral density of the time series.
friedrich_coefficients(x, param)	Coefficients of polynomial h(x), which has been fitted to
has_duplicate(x)	Checks if any value in x occurs more than once

Table C.1 continued from previous page

has_duplicate_max(x)	Checks if the maximum value of x is observed more than once
has_duplicate_min(x)	Checks if the minimal value of x is observed more than once
index_mass_quantile(x, param)	The relative index i of time series x where q% of the mass of x lies left of i.
kurtosis(x)	Returns the kurtosis of x
large_standard_deviation(x, r)	Does time series have large standard deviation?
last_location_of_maximum(x)	Returns the relative last location of the maximum value of x.
last_location_of_minimum(x)	Returns the last location of the minimal value of x.
lempel_ziv_complexity(x, bins)	Calculate a complexity estimate based on the Lempel-Ziv compression algorithm.
length(x)	Returns the length of x
linear_trend(x, param)	Calculate a linear least-squares regression for the values of the time series versus the sequence from 0 to length of the time series minus one.
linear_trend_timewise(x, param)	Calculate a linear least-squares regression for the values of the time series versus the sequence from 0 to length of the time series minus one.
longest_strike_above_mean(x)	Returns the length of the longest consecutive subsequence in x that is bigger than the mean of x
longest_strike_below_mean(x)	Returns the length of the longest consecutive subsequence in x that is smaller than the mean of x
matrix_profile(x, param)	Calculates the 1-D Matrix Profile and returns Tukey's Five Number Set plus the mean of that Matrix Profile.
max_langevin_fixed_point(x, r, m)	Largest fixed point of dynamics
maximum(x)	Calculates the highest value of the time series x.
mean(x)	Returns the mean of x
mean_abs_change(x)	Average over first differences.
mean_change(x)	Average over time series differences.
mean_n_absolute_max(x, number_of_maxima)	The arithmetic mean of the n absolute maximum values of the time series.
mean_second_derivative_central(x)	Returns the mean value of a central approximation of the second derivative
median(x)	Returns the median of x
minimum(x)	Calculates the lowest value of the time series x.
number_crossing_m(x, m)	Calculates the number of crossings of x on m.
number_cwt_peaks(x, n)	Number of different peaks in x.

Table C.1 continued from previous page

number_peaks(x, n)	Calculates the number of peaks of at least support n in the time series x.
partial_autocorrelation(x, param)	Calculates the value of the partial autocorrelation function at the given lag.
percentage_of_reoccurring_datapoints_to_all_datapoints(x)	Returns the percentage of non-unique data points.
percentage_of_reoccurring_values_to_all_values(x)	The percentage of values that are present in the time series more than once.
permutation_entropy(x, tau, dimension)	Calculate the permutation entropy.
quantile(x, q)	Calculates the q quantile of x.
query_similarity_count(x, param)	This feature calculator accepts an input query subsequence parameter, compares the query (under z-normalized Euclidean distance) to all subsequences within the time series, and returns a count of the number of times the query was found in the time series (within some predefined maximum distance threshold).
range_count(x, min, max)	Count observed values within the interval [min, max].
ratio_beyond_r_sigma(x, r)	Ratio of values that are more than r * std(x) away from the mean of x.
ratio_value_number_to_time_series_length(x)	Returns a factor which is 1 if all values in the time series occur only once, and below one if this is not the case.
root_mean_square(x)	Returns the root mean square (rms) of the time series.
sample_entropy(x)	Calculate and return sample entropy of x.
set_property(key, value)	Returns a decorator that sets the property key of the function to value
skewness(x)	Returns the sample skewness of x
spkt_welch_density(x, param)	The cross power spectral density of the time series x at different frequencies.
standard_deviation(x)	Returns the standard deviation of x
sum_of_reoccurring_data_points(x)	The sum of all data points, that are present in the time series more than once.
sum_of_reoccurring_values(x)	The sum of all values, that are present in the time series more than once.
sum_values(x)	Calculates the sum over the time series values
symmetry_looking(x, param)	Boolean variable denoting if the distribution of x looks symmetric.
time_reversal_asymmetry_statistic(x, lag)	Returns the time reversal asymmetry statistic.
value_count(x, value)	Count occurrences of value in time series x.
variance(x)	Returns the variance of x
variance_larger_than_standard_deviation(x)	Is variance higher than the standard deviation?
variation_coefficient(x)	Returns the variation coefficient (standard error / mean) of x.